RESOURCE-BOUNDED INFORMATION ACQUISITION AND LEARNING

A Dissertation Presented

by

PALLIKA H. KANANI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Department of Computer Science

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **MAY 2012**	2. REPORT TYPE	3. DATES COVERED

4. TITLE AND SUBTITLE **Resouce-Bounded Information Acquisition And Learning**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Massachusetts Amherst ,Amherst,MA,01003**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

The original document contains color images.

14. ABSTRACT

In many scenarios it is desirable to augment existing data with information ac- quired from an external source. For example, information from the Web can be used to fill missing values in a database or to correct errors. In many machine learning and data mining scenarios, acquiring additional feature values can lead to improved data quality and accuracy. However, there is often a cost associated with such information acquisition, and we typically need to operate under limited resources. In this thesis, I explore different aspects of Resource-bounded Information Acquisition and Learning.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **146**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

RESOURCE-BOUNDED INFORMATION ACQUISITION AND LEARNING

A Dissertation Presented

by

PALLIKA H. KANANI

Approved as to style and content by:

Andrew McCallum, Chair

David Jensen, Member

Shlomo Zilberstein, Member

Iqbal Agha, Member

Prem Melville, Member

Lori A. Clarke, Department Chair
Department of Computer Science

ACKNOWLEDGMENTS

My final project for the Natural Language Processing class at New York University was called ‘FindGuru,’ a system for enabling a student in any part of the world find the right research advisor - the right ‘Guru.’ Soon after that project (and partially, *because* of it), I found Andrew McCallum, a ‘Guru’ in true sense of the word. He taught me how to do research; how to think about it, write about it and talk about it. He showed me how to select good research problems, combine theory and practice, and navigate various fields of ideas. He also showed me how to achieve balance while being extremely passionate about research. Most importantly, like a true ‘Guru’, he never lost faith in me, even during the times when I was unsure of my path. I will always be grateful for his support.

I am thankful for the useful feedback from my committee members: David Jensen, Shlomo Zilberstein, Iqbal Agha and Prem Melville. Prem has also been a wonderful mentor and collaborator; and he, along with my other internship mentors, Krysta Svore and David Gondek helped me gain valuable research experience outside the university. During these internships, I had a great time working with my managers, fellow interns and co-workers. I would like to thank my collaborators: my ‘lab mentor’, Aron Culotta for early, hands-on training, Chris Pal for his patience through my first publication, Ramesh Sitaraman for his amazing support during the synthesis project, and Micheal Wick, Rob Hall and Shaohan Hu for fun experimental work. Thanks to Adam Saunders for answering infinite number of questions on Rexa.

I would like to sincerely thank Avrim Blum, Sridhar Mahadevan, Arnold Rosenberg, Richard Lawrence, Claudia Perlich, Andrew McGregor, Gideon Mann, Gerald Tesauro, Laura Dietz and Siddharth Srivastava for their time and helpful discussions.

I thank all anonymous reviewers of my papers for taking the time to provide useful feedback. All past and present members of IESL (Information Extraction and Synthesis Lab) have been great friends and collegues, and I thank them all for encouragement, ideas and impromptu discussions. A special thanks to Kedar Bellare, Greg Druck, and Micheal Wick for all the help on my projects.

I have also been very fortunate to have studied under wonderful professors throughout my academic life, who set high standards, and encouraged me to pursue knowledge. I thank them all for their dedication, and aspire to meet those high standards. The Computer Science department at UMass, Amherst has one of the most friendly and conducive environments for learning. Some of the people who went out of their way to make my life easier are: Kate Moruzzi, who can solve any problem, Glenn Stowell, Andre Gauthier, Dan Parker, Sharon Mallory, Late Pauline Hollister, Leeanne Leclerc, Barbara Sutherland, Dianne Muller and the wonderful staff of CSCF, who always handled my panic situations efficiently.

I have really been blessed by the support of innumerable people through the journey of my PhD. I wish I could thank each one individually. I really couldn't have made it without the constant encouragement from Pooja Jain, Kapil Jain, Reshma Varghese, Kavita Kukday-Deb, Sanchayeeta Borthakur, Siddharth Srivastava, Hema Raghavan, Aruna Balasubramanian, Lisa Friedland, Abilash Menon, Sandeep Menon, Upendra Sharma, Ruchita Tiwary, Niketa Jani and the never ending patience from my awesome room mates Anu Akella, Marshneil Deshmukh, Shweta Jain, Ujjwala Dandekar, Mandeep Kaur, Prakruti Desai, Shruti Vyas, and Meredith Nelson.

I am grateful to my mother-in-law, Vanita Madhwani, brother-in-law, Jitesh Madhwani, and the family for being extremely supportive of my work. My husband, Lokesh Madhwani has been a constant companion through the ups and downs of graduate life, and a rock that I could lean on at all times. I thank him for every sacrifice, big and small, that he made so I could finish, and also for making me laugh.

My brother, Harin Kanani has been a role model all my life and the family a source of joy. Finally, I can not thank my parents, Haridas and Beena Kanani enough for helping me become the person that I am, for teaching me the value of honesty, integrity, intellectual curiosity and hard work, inspiring me to aim high and believe in myself, and always supporting every decision I ever made.

This research draws on data provided by the University Research Program for Google Search, a service provided by Google to promote a greater common understanding of the web. I would also like to acknowledge the various funding sources that supported me throughout my graduate studies. This work was supported in part by the Center for Intelligent Information Retrieval, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grants #IIS-0326249 and NSF medium IIS-0803847, the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010, and AFRL #FA8750-07-D-0185, Microsoft Research under the Memex funding program, DoD contract #HM1582-06-1-2013, Lockheed Martin through prime contract #FA8650-06-C-7605 from the Air Force Office of Scientific Research. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

ABSTRACT

RESOURCE-BOUNDED INFORMATION ACQUISITION AND LEARNING

MAY 2012

PALLIKA H. KANANI

B.E., UNIVERSITY OF MUMBAI

M.S., NEW YORK UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew McCallum

In many scenarios it is desirable to augment existing data with information acquired from an external source. For example, information from the Web can be used to fill missing values in a database or to correct errors. In many machine learning and data mining scenarios, acquiring additional feature values can lead to improved data quality and accuracy. However, there is often a cost associated with such information acquisition, and we typically need to operate under limited resources. In this thesis, I explore different aspects of Resource-bounded Information Acquisition and Learning.

The process of acquiring information from an external source involves multiple steps, such as deciding what subset of information to obtain, locating the documents that contain the required information, acquiring relevant documents, extracting the specific piece of information, and combining it with existing information to make useful decisions. The problem of Resource-bounded Information Acquisition (RBIA) involves saving resources at each stage of the information acquisition process. I explore four special cases of the RBIA problem, propose general principles for efficiently acquiring external information in real-world domains, and demonstrate their effectiveness using extensive experiments. For example, in some of these domains I show how interdependency between fields or records in the data can also be exploited to achieve cost reduction. Finally, I propose a general framework for RBIA, that takes into account the state of the database at each point of time, dynamically adapts to the results of all the steps in the acquisition process so far, as well as the properties of each step, and carries them out striving to acquire most information with least amount resources.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

PUBLICATIONS

Some of the work presented in this dissertation has been previously published through following papers:

- Kanani, P. and McCallum, A. “Selecting Actions for Resource-bounded Information Extraction using Reinforcement Learning”, In the proceedings of WSDM 2012.

- Kanani, P. and McCallum, A. “Learning to Select Actions for Resource-bounded Information Extraction”, UMass TechReport UM-CS-2011-042, 2011.

- Kanani, P., McCallum A. and Hu S., “Resource-bounded Information Extraction: Acquiring Missing Feature Values On Demand”, In the proceedings of PAKDD 2010.

- Kanani, P., McCallum, A., and Sitaraman, R., Towards Theoretical Bounds for Resource-bounded Information Gathering for Correlation Clustering, UMass TechReport UM-CS-2009-027

- Kanani, P. and Melville, P., “Prediction-time Active Feature-value Acquisition for Customer Targeting”, NIPS 2008 Workshop on Cost Sensitive Learning.

- Kanani, P. and McCallum, A., “Efficient Strategies for Improving Partitioning-Based Author Coreference by Incorporating Web Pages as Graph Nodes,” AAAI 2007 Workshop on Information Integration on the Web (IIWEB 07), pp. 38-43. Also appeared as a poster in NESCAI 2007.

- Culotta, A., Kanani, P., Hall, R., Wick, M. and McCallum, A., "Author Disambiguation using Error-driven Machine Learning with a Ranking Loss Function," AAAI 2007 Workshop on Information Integration on the Web (IIWeb 07).

- Kanani, P. and McCallum, A., "Resource-bounded Information Gathering for Correlation Clustering," in the Proceedings of COLT 2007, Open Problems Track, LNAI 4539, pp. 625-627, 2007.

- Kanani, P., McCallum, A. and Pal, C., "Improving Author Coreference by Resource-bounded Information Gathering from the Web," in the Proceedings of IJCAI 2007, pp. 429-434, 2007.

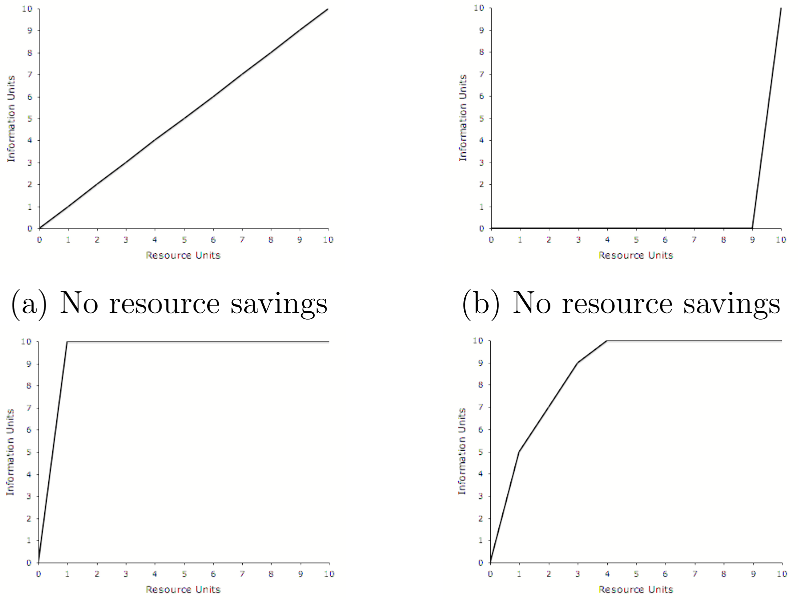
CHAPTER 1
INTRODUCTION

1.1 Problem Overview and Motivation

Information is a valuable commodity and in many scenarios, we would like to acquire additional information from an external source. For example, we can increase the utility of most databases by filling in missing, incomplete or uncertain information. Accuracy of most data mining applications can be improved by acquiring additional features and instances. The source of this additional information can be an external, structured database, a semi-structured or unstructured document corpus, or an extremely large, heterogenous corpus, such as the Web. However, there is often a significant cost associated with gathering and integrating this additional information. It is not desirable, and even prohibitive, for example, to purchase every available database or crawl and parse every page on the web. The resources required for this task may include computer processing, storage space, network bandwidth, database schema mapping, as well as monetary, time, human, and administrative costs. *Resource-bounded Information Acquisition* (RBIA) is the process of efficiently allocating and targeting expensive or scarce resources to find, acquire and integrate the most beneficial additional information.

Consider the process of efficiently acquiring external information. The first step is often deciding what information to obtain, since some information may be more valuable than other, motivating us to prioritize the acquisition of those pieces that would help achieve our final goal. In some cases, the required information may be readily available in a suitable form on the external source. However, in most scenarios, it may be embedded in a structured, semi-structured, or unstructured document, which, in turn is part of a large corpus. In such scenarios, we need to request the external source for location of the document, via a search interface. After locating relevant documents that potentially contain the required information, we need to transfer them to our local computing device, before we can process them. If the external information is not in a structured form, we need to process these documents to extract the specific piece of information that we are interested in. Also, as new information arrives, we need to combine it with existing information, so as to make decisions about our confidence in the values of the database entries. Resource-bounded Information Acquisition considers steps to reduce efforts at each of these stages of information gathering process. In this thesis, I propose a broad RBIA framework, and explore various special cases thereof.

In practice, the amount of resources we can save is application-specific, and there is a wide spectrum of applications that provide opportunities to save information gathering resources. Consider a toy-world information gathering setup. Let us assume that we have 10 units of resources of some type, and 10 units of information to acquire. Figure 1.1 shows examples of information gathering scenarios that represent different degrees of resource saving opportunities. At one end of the spectrum are examples 1.1(a) and (b), in which absolutely no resource-saving is possible. In 1.1(a), each step in the information gathering process is independent of each other, and helps acquire exactly equal amount of information. Hence, we must use all 10 units of resources to acquire the required information. The example 1.1(b) represents a case, in which we only need one unit of the resource to acquire all the information. However, before we can carry out the step that acquires this information, we must use 9 units of resources on steps that must precede it. On the other end of the spectrum is the case, in which we only need one unit of resource to acquire all information, as shown in 1.1(c). This provides an opportunity for significant resource savings, provided we know the correct ordering of steps. Most real world applications lie somewhere between these two extreme scenarios. 1.1(d) shows such a realistic scenario, in which we have the opportunity to acquire all or most of the information using only a fraction of the total resources.



(a) No resource savings

(b) No resource savings

(c) Significant resource savings

(d) Some resource savings

Figure 1.1. Example Information Gathering Scenarios

The application domains I study in this thesis cover a broad range of the RBIA problem spectrum, and the methods I propose spread across various aspects to be considered when designing solutions for RBIA problems. In some instantiations of the RBIA framework, we save resources by focusing on only a few stages of information acquisition process, such as selecting a subset of the input instances for which to acquire information. Other instantiations provide a broader view, by saving resources on each stage of the acquisition process. In some domains, we exploit the interdependency within the input data, whereas in other domains, we develop more general methods applicable even when the input instances are non-relational. The goal of this thesis is to develop a comprehensive framework for Resource-bounded Information Acquisition, that takes into account the state of the database, the results of all the steps in the acquisition process so far, as well as the properties of each step, and carries them out so as to acquire most information with least resources.

1.2 RBIA General Problem Definition

We are given a database with a set of missing or uncertain values, and access to an external source of information. We define the following four types of resource-consuming, information acquisition actions:

- **Query:** Issue information or search request to the external source
- **Download:** Transfer a document from the external source to a local device
- **Extract:** Process a document to extract the required piece of information
- **DB-Inference:** Use the information from extraction or available within the database to adjust database values.

The problem of Resource-bounded Information Acquisition (RBIA) is to select the 'best' among all available actions at each point of time, so as to acquire most information, using least amount of resources.

1.3 Information Acquisition Actions

We can view the database as a collection of variables with missing, or uncertain values. Each of the information acquisition actions defined above help obtain more accurate values for these variables. Let us examine the nature of these actions, as well as the types of resources they consume in further detail.

The *query* action consists of issuing a request to the external information source for returning the required information, or the location thereof (e.g., a web-search API). For each variable with missing or uncertain value, there may be multiple types of queries, some more effective than others, that can aid in obtaining information. This leads to a large number of possible queries to be issued. However, there may be several types of resources consumed in issuing such queries. For example, issuing all types of queries for every instance, or for every feature of each instance may be time consuming. There may be a restriction on the number of queries allowed by the search interface (as in the case of web queries). There may also be monetary cost involved, if buying the information from an external source. Selecting a *query* action, therefore, involves selecting the input variable for which to acquire information, as well as the type of query used to acquire it. Hence, we can view the *query* action to have the following parameters: an input instance, a specific feature or field of an input instance, and the type of query to be issued.

In many information acquisition scenarios, the required information resides on the external source in the form of documents. We need to transfer such documents to our local computing device, before we can access the required information. This task is carried out through a *download* action. Based on the nature of the information interface provided by the external source, we may be faced with the option of acquiring a large number of documents provided as a result of a *query* action (e.g., web-search results). The resources typically consumed in this process are network bandwidth and storage space. In some cases, there may be a monetary cost associated with obtaining each document. Hence, we must transfer only a subset of these documents to conserve resources.

When acquiring information from semi-structured or unstructured documents, we may need to apply sophisticated and computationally expensive methods for extracting the specific piece of information required. An *extract* action consists of performing extraction on the downloaded document to obtain the required piece of information and using it to fill the slot in the original database. Note that even after deciding to download a document, we may find from preliminary examination that the document is not suitable for extraction. In such scenarios, we may decide not to perform an *extract* action.

By taking the view of database as a collection of variables, we can define our information goals in terms of finding their true values. We can use information already available in the database, along with that acquired from the external source to infer missing values of the variables. In many scenarios, there may be uncertainty associated with existing values in the database, and the additional information may serve to reduce it. We call the process of using all available information to determine the values of database variables as a *db-inference* action. In some cases, for example, if the database variables are i.i.d., the *db-inference* may consist of evaluating a confidence measure on their values. In the case when input data is relational in nature, *db-inference* may take a more complex form. In the following chapters, we will see examples of each of these scenarios.

Before selecting an action to perform at each time step, we need to consider several factors. We need to take into account the current state of the database, such as the number of slots filled and the uncertainty about them. We need to take into account the context provided by the intermediate results of all the actions so far, such as the results of the queries, documents that are not yet downloaded and processed. Even if this context is not yet in the database, these intermediate results can provide valuable information for deciding which action to select. Finally, we also need to consider the properties of the candidate action itself, before selecting it.

It is important to note that for a given RBIA system, some of these actions may be non-existent or trivial. Also, they may interact with each other in interesting ways. Consider the following scenario: the system may face a choice between running a really complex, expensive inference on the data that already exists in the database and running an inexpensive query to acquire it from an external source. Another scenario is, after performing some *db-inference*, we learn that we know the value of a variable with high confidence, and hence decide not to issue the corresponding query. We will see examples of such scenarios in this thesis.

Another issue to note is that in many cases, all information acquisition actions may not be available initially. Some actions may be created, or instantiated as a result of other actions. For example, in the case of extracting information from the Web, the *query* actions can be initialized at the beginning of the task because we know which instances have missing fields, and the types of queries that can be used; but *download* actions and *extract* actions are generated dynamically and added to the list of available actions. That is, after a *query* action is performed, the *download* action corresponding to each of the search results is generated. Similarly, after a web page is downloaded, the corresponding *extract* action is generated. At each time point, only the actions that are instantiated can be considered as alternative valid actions to be performed.

1.4 The RBIA Solution Landscape

On a broad level, Resource-bounded Information Acquisition (RBIA) is a multi-dimensional problem and we need to consider following issues while designing the classes of solutions to a given RBIA problem. One way to view RBIA is as a subset selection problem, in that we can either choose to select a subset of input instances for which to obtain information, or select a subset of information to acquire. Another dimension to study is myopic vs. non-myopic information acquisition. The designed solution may be static, i.e. follow a pre-determined plan, or dynamic, and adapt to the changing information scenario. Furthermore, we may be given a fixed resource budget to operate under, or we need to design a solution, such that we get the best possible resource-utilization at each step of the information gathering process. If the information acquired is used for a machine learning application, we also need to consider if it would be used at train or test time. Finally, the relational nature of data poses interesting questions and opportunities for optimal resource allocation. In this thesis, I present example domains and corresponding solutions that explore a large portion of this multi-dimensional space, and propose a framework that is general enough to design a good solution, based on the given RBIA problem definition.

1.5 Thesis Outline

- **Related Work**(Chapter 2). I describe how my work is uniquely positioned between other approaches in this area.

- **Resource-bounded Information Gathering for Graph Partitioning** (Chapter 3). I present a special case of RBIA, in which the input instances are interrelated, and we need to select a subset of queries to issue. I formulate the problem of selectively acquiring additional information in the context of graph partitioning for entity resolution. The two approaches presented to improve the quality of the underlying graph by using external information are: improving the accuracy of edge weights and adding new nodes, so as to aid in better partitioning of the data. I propose multiple criteria for selecting edges in the graph, such that obtaining more information about them leads to a high overall impact on the partitioning. I empirically demonstrate effectiveness of the expected entropy based approach for edge selection, which takes into account the global impact of new information, as opposed to local uncertainty. I also propose methods for effectively incorporating additional nodes in the graph and discuss their application. Finally, I describe a general, theoretical open problem that stems out of this work.

- **Test-time Active Information Acquisition** (Chapter 4). Next, I present another instantiation of RBIA, in which the focus is again on selecting a subset of instances for which to acquire external information, i.e. a subset of queries, but the input instances are not interdependent. This work generalizes the methods presented for the case of information acquisition for graph partitioning for the i.i.d. input case. Building on previous work in active feature acquisition at train time, I present methods for effectively selecting instances for acquiring additional features at test time. Class labels are useful in evaluating the value of acquiring features, and they are available at train time, but not at test time. In this work, we show how to circumvent this problem, which is one of the key contributions. Extensive experimental results on customer targeting applications confirm that our proposed approaches can effectively select instances for which it is beneficial to acquire more information to classify them better, as compared to acquiring additional information for the same number of randomly sampled instances.

- **Exploiting Interdependency for Resource-bounded Information Extraction** (Chapter 5). In this instantiation of RBIA, I expand my focus to include all steps of the information acquisition process, instead of focusing only on selecting the input queries. I consider the case, in which the required information must be extracted from web documents, and introduce the problem of Resource-bounded Information Extraction (RBIE). The main contribution of this work is the idea of information propagation through the network of input instances for reducing uncertainty, which in turn leads to reduction in required resources for acquiring new information. However, the majority of resource savings in this domain come from exploiting interdependency in the input data to improve resource utilization, hence, it is not generalizable to all domains. Another drawback of this method is that it fails to capture the interactions between various information acquisition steps, so as to order them effectively.

- **Resource-bounded Information Extraction for the Web as an MDP** (Chapter 6). I propose a general framework for RBIE that overcomes the drawbacks of previous methods. It does not depend on the relational nature of the input data, making it more generally applicable, and considers all types of available steps simultaneously for effective information acquisition. It also adapts the information gathering approach dynamically, based on the results of the steps so far, making it a flexible and effective approach. I use Markov Decision Process for targeted, resource-bounded information extraction from the Web. I also demonstrate the effectiveness of temporal difference q-learning in learning to make sequential decisions from data for two example tasks, and compare it to an online, error-driven algorithm called SampleRank, along with strong baselines. The proposed methods are able to obtain a large portion of the total information, using only a fraction of resources.

- **Future Work** (Chapter 7). I describe directions for extending or improving the proposed framework in the future.

1.6 Thesis Contributions

Here are the specific contributions of my thesis:

- I provide a specific definition of an important class of problems, called Resource-bounded Information Acquisition (RBIA). I believe that this problem formulation encompasses different aspects of the domains in which they occur, and facilitates development of useful solutions.

- In this thesis, I ask and answer the following question. Given naturally occurring problem domains with incomplete information, is it possible to significantly reduce the amount of resources required to acquire additional, external information? I demonstrate using various empirical evaluations, that we can indeed achieve a large fraction of the total benefit from new information, by only using a small fraction of the resources. For instance, in the task of extracting faculty information from the Web, we are able to achieve 88.8% of the final F1 value (that we would have been able to achieve by using all possible resource-consuming actions), by only using 8.6% of the total actions.

- I propose a novel framework for solving RBIA problems. I demonstrate the effectiveness of this framework on four problem domains, and four special cases of the framework showing the generality and applicability of the proposed framework.

CHAPTER 2

RELATED WORK

Learning and acquiring information under resource constraints has been studied in various forms. In this chapter, I describe different aspects of this problem and how my work is uniquely positioned between them. I start by discussing classical work in information extraction tasks, followed by methods aimed at large scale information extraction from the Web. Next, I discuss general active information acquisition methods, and finally more theoretical work in resource-bounded reasoning.

2.1 Information Extraction From the Web

In the traditional information extraction settings, we are usually given a database schema, and a set of unstructured or semi-structured documents. The goal of the system is to automatically extract records from these documents, and fill in the values in the given database. These databases are then used for search, decision support and data mining. In recent years, there has been much work in developing sophisticated methods for performing information extraction over a closed collection of documents [35, 47]. Several different approaches have been proposed for different phases of information extraction task, such as segmentation, classification, association and coreference. Most of these proposed approaches make extensive use of statistical machine learning algorithms, which have improved significantly over the years. However, only some of these methods remain computationally tractable as the size of the document corpus grows. In fact, very few systems are designed to scale over a corpus as large as, say, the Web [28, 97].

Early work on extracting information from the Web was conducted by Brin [12] and Etzioni et al. [29]. Rennie and McCallum [73] built a web spider using Reinforcement Learning, which served as a foundation for some of the ideas presented in Chapter 6. There are some large scale systems that extract information from the web. Among these are KnowItAll [28, 30], InfoSleuth [70] and Kylin [93]. The goal of the KnowItAll system is a related, but different task called, "Open Information Extraction." In Open IE, the relations of interest are not known in advance, and the emphasis is on discovering new relations and new records through extensive web access. In contrast, in our task, what we are looking for is very specific and the corresponding schema is known. The emphasis is mostly on filling the missing fields in known records, using resource-bounded web querying. Hence, OpenIE and RBIE frameworks have very different application domains. InfoSleuth focuses on gathering information from given sources, and Kylin focuses only on Wikipedia articles. Among other systems that aim to extract entity names and relations from the web are, NELL [13], SOFIE [82], DBLife [21], Cyclex [14], xCrawl [79], Factzor [91], and WebSets [19]. These systems also do not aim to exploit the inherent dependency within the database for maximum utilization of resources, as we do in Chapter 5. Gatterbauer [31] provides interesting theoretical insights into exploiting redundancy on the Web for obtaining the required coverage of data.

Agichtein and Gravano [1] develop an automatic query-based technique to retrieve documents useful for the extraction of user-defined relations from large text databases and improve the efficiency of the extraction process by focusing only on promising documents. Similarly, Agrawal et al. [2] tackle "ad-hoc" entity extraction task, where entities of interest are constrained to be from a list of entities that is specific to the task. They propose an approach that uses an inverted index on the documents to only process relevant documents. Huang et al. [38] propose a prioritization approach where candidate pages from the corpus are ordered according to their expected contribution to the extraction results and those with higher estimated potential are extracted earlier. The RBIE framework presented in this thesis provides a more general and adaptable approach, with not just document filtering and ranking, but a sequential decision making process for better resource utilization. Elliasi-Rad [26] explored the problem of building an information extraction agent, but did not address the problem of acquiring specific missing pieces of information on demand.

The Knowledge Base Population (KBP) Track, which is part of the Text Analysis Conference focuses on related tasks. The emphasis in these tasks is, however on filling slots in Wikipedia info boxes, and not general purpose, targeted information extraction tasks. The Information Retrieval community is rich with work in document relevance (TREC). However, traditional information retrieval solutions can not directly be used, since we first need to automate the query formulation for our task. Also, most search engine APIs return full documents or text snippets, rather than specific values.

A family of methods closely related to RBIE is question answering systems [51]. These systems do retrieve a subset of relevant documents from the web, along with extracting a specific piece of information. However, they target a single piece of information requested by the user, whereas we target multiple, interdependent fields of a relational database. They formulate queries by interpreting a natural language question, whereas we formulate and rank them based on the utility of the information within the database. They do not address the problem of selecting and prioritizing instances or a subset of fields to query. This is why, even though some of the components in our system may appear similar to that of QA systems, their functionalities differ. The semantic web community has also been working on similar problems, but the focus is not targeted information extraction. A few systems have been developed for extracting researcher information from the Web [84, 96, 52, 68, 66, 67], some of which use regular expressions, and others use more formal models like Conditional

Random Fields for extraction, but none of them focus on sequential decision making for resource optimization, as described in chapter 6.

2.2 Active Information Acquisition

Learning and acquiring information under resource constraints has been studied in various forms. For a comprehensive overview of various information acquisition scenarios, please refer to [40]. Settles [78] provides a good survey of active learning literature. We first look at different information acquisition scenarios at training time. The most common scenario is *active learning* [15, 85, 74], which assumes access to unlabeled instances with complete feature values and attempts to select the most useful instances for which to acquire class labels while training. The next scenario is *active feature acquisition*, which explores the problem of learning models from incomplete instances by acquiring additional features[62, 61, 71]. The general case of acquiring randomly-missing values in the instance-feature matrix is addressed in [63, 64]. Our work, as described in chapter 4 builds on these ideas. More recent work [80, 23, 24] deals with learning models using noisy labels. A related problem, proactive learning [95], is a generalized form of active learning where the learner must reach out to multiple oracles exhibiting different costs and reliabilities. Attenberg et al. [4] introduce the problem of active inference, in which human labels are requested for inference with a limited labeling budget. The idea of labeling features, instead of labels has been studied under the generalized expectation criteria by Druck et al. [25], and Attenberg et al. [4].

Under the "budgeted descriminative attribute learning" (BDAL) scenario [53], all of the labels are given, the total cost to be spent towards acquisitions is determined a priori, and the task is to identify the best set of attribute values to be acquired for this cost. This model takes into account the dependencies among attributes as well as the dependencies between the attributes and the labels. Also, different attributes can have different costs. Another budgeted learning scenario is the "budgeted distribution learning" (BDL) framework proposed in [50, 65, 56]. The main goal of BDL framework is to build a generative model, as opposed to a discriminative model of BDAL, and does not distinguish between attributes and labels. The BDL work is also related to the "interventional active learning" (IAL) framework [86]. Here, the learner sets the values of a fixed set of features (interventions), and then acquires the values of the remaining instances at a fixed cost. Esmeir et al. [27] study anytime algorithms for producing tree-based classifiers that can make accurate decisions within a strict bound on testing costs. Turney [87] created a taxonomy of the different types of cost that are involved in inductive concept learning.

There has also been some work on prediction-time AFA, but the focus has been on selecting a subset of features to acquire, rather than selecting a subset of instances for which to acquire the features. For example, Bilgic et al.[7] exploit the conditional independence between features in a Bayesian network for selecting a subset of features. Similarly, Sheng et al. [81] aim to reduce acquisition cost and misclassification under different settings, but their approach also focuses on selecting a subset of features. Wu et al. [94] study the problem of online streaming feature selection, in which the size of the feature set is unknown, and not all features are available for learning while leaving the number of observations constant. In this problem, the candidate features arrive one at a time, and the learner's task is to select a 'best so far' set of features from streaming features. Krause et al. [45] apply the theory of value of information, but their method is mostly restricted to chain graphical models. Golovin et al. [33] tackle the problem of Bayesian active learning with noise, to adaptively select from a number of expensive tests in order to identify an unknown hypothesis sampled from a known prior distribution. Gatterbauer [32] presents an abstract model of information acquisition from redundant data, and characterizes the process of randomized sampling from biased information.

The interdependency within the data set is often conveniently modeled using graphs, but it poses interesting questions about selection of instances to query and propagating uncertainty through the graph [41]. Chapter 3 describes the case in which the test instances are not independent of each other, and we study the impact of acquisition in the context of graph partitioning. Similar problems are addressed in [8, 72]. Bilgic et al. [9] introduce a novel active learning algorithm for classification of network data, whereas Kuwadekar et al. [46] combines semi-supervised learning and relational resampling for active learning in network domains. Macskassy [54] also exploits graph structure in the data to select candidates for labeling. Nath and Domnigos [69] combine graphical models with first order logic to provide a general language for relational decision theory. Ideas from other fields, such as graph theory [20] and circuit design [55] can also be borrowed in this context. The general RBIE framework described in chapter 5 aims to leverage these methods for both train and test time for optimization of query and instance selection, depending on the application scenario.

2.3 Resource-bounded Reasoning

Another body of related work is in the area of preference elicitation, which is the task of gathering the preference or utility function of specific users. Boutilier [11] argues that determining which information to extract from a user is itself a sequential decision problem, balancing the amount of elicitation effort and time with decision quality, and hence formulates this problem as a partially-observable Markov decision process (POMDP). This idea is similar to our MDP formulation for Resource-bounded Information Extraction. Connections between concept learning and preference elicitation have been explored by Blum et al. [10]. More recently, Boutilier et al. [16, 17] presented a regret based model for utility elicitation that allows users to define their own subjective features over which they can express their preferences. Bardak et al.

[6] present a similar task of scheduling a conference based on incomplete data about available resource and scheduling constraints, and describe a procedure for automated elicitation of additional data. More recently, Viappiani et al. [88] present an analysis of set-based recommendations in Bayesian recommender systems, and show how to generate myopically optimal or near-optimal choice queries for preference elicitation.

Knoblock et al. [44] introduced the idea of using planning for information gathering, followed by the development of resource-bounded reasoning techniques by Zilberstein et al. [99]. Value of information, as studied in decision theory, measures the expected benefit of queries [100, 37]. Resource-bounded reasoning studies the trade offs between computational commodities and value of the computed results [99]. Grass and Zilberstein [34] present a system for autonomous information gathering that consider time and monetary resource constraints. Their system uses an explicit representation of the user's decision model, which is not the focus of our work. However, the Expected Utility approach described in this thesis follows these ideas. As proposed RBIE framework develops further, more formal models of cost and utility can be applied for better performance with respect to the user's utility function. Lesser et al. [48] also build a planning based resource-bounded information gathering agent, that locates, retrieves, and processes information to support a decision process. This work provides interesting insights into the architecture of building such a system, and address some aspects of information gathering that we do not. However, we believe that the RBIE framework is more flexible in terms of being able to define general purpose actions, as information acquisition scenarios change.

Arnt et al. [3] apply decision theoretic ideas for the problem of sequential time and cost sensitive classification. Kapoor et al. [42] provide a theoretical analysis of budgeted learning when the learner is aware of cost constraints at prediction-time. This work is followed up [43], with information acquisition strategies that bridge the gap between training and test time. The idea of value of information for resource-bounded computation has also been applied in various other domains, for example, Vijayanarasimhan et al. [89] apply it in computer vision for visual recognition and detection. This demonstrates the generality and importance of ideas considered in this thesis.

CHAPTER 3

RESOURCE-BOUNDED INFORMATION GATHERING FOR AUTHOR COREFERENCE

3.1 Introduction

Machine learning and web mining researchers are increasingly interested in using search engines to gather information for augmenting their models [28, 57, 22]. Some of these methods rely on issuing queries to a web search engine API, such as Google to acquire the required information. For a given problem, there can be multiple types of queries issued, some more useful than others. In many real world applications, there may be a large number of input instances, and issuing even a single type of query for all input instances maybe extremely expensive. However, we may be able to exploit the fact that information for some input instances may be more valuable than others in achieving improved accuracy on the final task. This gives rise to the problem of efficiently selecting the queries that would provide the most benefit. We refer to this problem as *Resource-bounded Information Gathering (RBIG) from the Web.*

Let us examine this problem in the domain of entity resolution. Given a large set of entity names (each in their own context), the task is to determine which names are referring to the same underlying entity. Often these coreference merging decisions are best made, not merely by examining separate pairs of names, but relationally, by accounting for transitive dependencies among all merging decisions. Following previous work, we formulate entity resolution as graph partitioning on a weighted, undirected, fully connected graph, whose vertices represent entity mentions, and edge weights represent the probability that the two mentions refer to the same entity. In this chapter, we explore a relational, graph-based approach to resource-bounded information gathering, i.e., the *db-inference* action from the RBIA framework takes the form of graph partitioning.

The specific entity resolution domain we address is research paper author coreference. The vertices in our coreference graphs are citations, each containing an author name with the same last name and first initial. Coreference in this domain is extremely difficult. Although there is a rich and complex set of features that are often helpful, in many situations they are not sufficient to make a confident decision. Consider, for example, the following two citations both containing a "D. Miller."

- Mark Orey and David Miller, Diagnostic Computer Systems for Arithmetic, Computers in the School, volume 3, #4, 1987

- Miller, D., Atkinson, D., Wilcox, B., Mishkin, A., Autonomous Navigation and Control of a Mars Rover, Proceedings of the 11th IFAC Symposium on Automatic Control in Aerospace, pp. 127-130, Tsukuba, Japan, July 1989.

The publication years are close; and the titles both relate to computer science, but there is not a specific topical overlap; "Miller" is a fairly common last name; and there are no co-author names in common. Furthermore, in the rest of the larger citation graph, there is not a length-two path of co-author name matches indicating that some of the co-authors here may have themselves co-authored a third paper. So there is really insufficient evidence to indicate a match despite the fact that these citations do refer to the same "Miller."

We present two different mechanisms for augmenting the coreference graph partitioning problem by incorporating additional helpful information from the web. In both cases, the *query* action consists of a web search engine query, which is formed by conjoining the titles from two citations. The first mechanism changes the edge weight between the citation pair by adding a feature indicating whether or not any web pages were returned by the query. In this case, we omit both, the *download* and *extract* actions, and replace them by a single piece of information (feature value) returned by the external source.

The second mechanism uses one of the returned pages (if any) to create an additional vertex in the graph, for which edge weights are then calculated to all the other vertices. In this case, we do not explicitly use an *extract* action. The additional transitive relations provided by the new vertex can provide significantly helpful information. For example, if the new vertex is a home page listing all of an author's publications, it will pull together all the other vertices that should be coreferent.

Gathering such external information for all vertex pairs in the graph is prohibitively expensive, however. Thus, methods that acknowledge time, space and network resource limitations, and effectively select just a subset of the possible queries are proposed. The RBIA solution in this case focuses on selecting the best *query* actions, based on their interaction with *db-inference.* The methods presented in section 3.5.2 also focus on selecting effective *download* actions.

In theory, it is extremely difficult to analyze the effect of changing the weight of a single edge on the overall clustering of the graph. In fact, we published this problem of deciding which query to select first, so as to optimize the use of resources, as an open theoretical problem [41].

3.2 General Problem Setup

Let $G_0(V_0, E_0)$ be a fully connected, weighted, undirected graph. Our objective is to partition the vertices in graph G_0 into an unknown number of M non-overlapping subsets. $E_0 = \{e_{ij}\}$ is the set of edges in G_0, where $e_{ij} =< v_i, v_j >$ is an edge whose weight $w_{ij} \propto p_{ij}$. Here, p_{ij} is the probability that vertices v_i and v_j belong to the same partition. We assume that p_{ij} is computed using a probabilistic model, with a set of existing pair-wise feature functions, $f_e(v_i, v_j)$. We now assume that we can acquire a new feature, $f_n(v_i, v_j)$ from an external source, as a result of a query involving information from vertices v_i and v_j, and that $f_n(v_i, v_j)$ may help improve our estimate of p_{ij}. Our first problem is, deciding the order in which we should select queries that correspond to edges in E_0, so as to obtain most benefit using least number of queries. Note that, for the query selection criteria to work effectively, our original estimate of p_{ij} needs to be at least better than random. If the initial estimate of p_{ij} is worse than random, or if the new feature $f_n(v_i, v_j)$ acquired from the external source is not informative, the methods proposed here may not be effective.

We also consider the scenario in which we can expand the graph G_0, by augmenting it with additional nodes, which represent documents obtained from an external source, such as the Web. Our assumption is that by partitioning this expanded graph, G_1, we may be able to achieve improved partitioning over the nodes in G_0, by imposing additional transitive relations. Our second problem is selecting appropriate queries for acquiring additional nodes in G_1, as well as finding a subset of these nodes to be included in G_1, so as to obtain most benefit with least amount of computational resources. In this case, we assume that there exist some external documents that potentially have strong affinity to multiple nodes. The methods proposed here may not be applicable in cases when such external information doesn't exist.

3.3 Conditional Entity Resolution Models

We are interested in obtaining an optimal set of coreference assignments for all mentions contained in our database. In our approach, we first learn maximum entropy or logistic regression models for pairwise binary coreference classifications. We then combine the information from these pairwise models using graph-partitioning-based methods so as to achieve a good global and consistent coreference decision. We use the term, "mention" to indicate the appearance of an author name in a citation and use x_i to denote mention $i = 1, \ldots, n$. Let y_{ij} represent a binary random variable that is true when mentions x_i and x_j refer to the same underlying author "entity." For each pair of mentions we define a set of l feature functions $f_l(x_i, x_j, y_{i,j})$ acting upon a pair of mentions. From these feature functions we can construct a local model given by

$$P(y_{i,j}|x_i, x_j) = \frac{1}{Z_x} \exp(\lambda_l f_l(x_i, x_j, y_{ij})), \tag{3.1}$$

where $Z_x = \sum_y \exp(\lambda_l f_l(x_i, x_j, y_{ij}))$. In [58] a conditional random field with a form similar to (3.1) is constructed which effectively couples a collection of pairwise coreference models using equality transitivity functions $f_*(y_{ij}, y_{jk}, y_{ik})$ to ensure globally consistent configurations. These functions ensure that the coupled model assigns zero probability to inconsistent configurations by evaluating to $-\infty$ for inconsistent configurations and 0 for consistent configurations. The complete model for the conditional distribution of all binary match variables given all mentions $\mathbf{x}$ can then be expressed as

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\Bigg(& \sum_{i,j,l} \lambda_l f_l(x_i, x_j, y_{ij}) + \\ & \sum_{i,j,k} \lambda_* f_*(y_{ij}, y_{jk}, y_{ik}) \Bigg), \end{aligned} \tag{3.2}$$

where $\mathbf{y} = \{\mathbf{y_{ij}} : \forall_{\mathbf{i,j}}\}$ and

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp\big(\sum_{i,j,l} \lambda_l f_l(x_i, x_j, y_{ij}) + \sum_{i,j,k} \lambda_* f_*(y_{ij}, y_{jk}, y_{ik})\big) \tag{3.3}$$

As in Wellner and McCallum [2002], the parameters λ can be estimated in local fashion by maximizing the product of Equation 1 over all edges in a labeled graph exibiting the true partitioning. When $f_l(x_i, x_j, 1) = -f_l(x_i, x_j, 0)$ it is possible to construct a new undirected and fully connected graph consisting of nodes for mentions, edge weights $\in [-\infty, \infty]$ defined by $\sum_l \lambda_l(x_i, x_j, y_{ij})$ and with sign defined by the value of y_{ij}. In our work here we define a graph in a similar fashion as follows.

Let $G_0 =< V_0, E_0 >$ be a weighted, undirected and fully connected graph, where $V_0 = \{v_1, v_2, ..., v_n\}$ is the set of vertices representing mentions and E_0 is the set of edges where $e_i =< v_j, v_k >$ is an edge whose weight w_{ij} is given by $P(y_{ij} = 1|x_i, x_j) - P(y_{ij} = 0|x_i, x_j)$ or the difference in the probabilities that that the citations v_j and v_k are by the same author. Note that the edge weights defined in this manner are in $[-1, +1]$. The edge weights in E_0 are noisy and may contain inconsistencies. For example, given the nodes v_1, v_2 and v_3, we might have a positive weight on $< v_1, v_2 >$ as well as on $< v_2, v_3 >$, but a high negative weight on $< v_1, v_3 >$. Our objective is to partition the vertices in graph G_0 into an unknown number of M non-overlapping subsets, such that each subset represents the set of citations corresponding to the same author.

We define our objective function as $\mathcal{F} = \sum_{ij} w_{ij} f(i, j)$ where $f(i, j) = 1$ when x_i and x_j are in the same partition and -1 otherwise.

Blum et al. provide two polynomial-time approximation schemes (PTAS) for partitioning graphs with mixed positive and negative edge weights [5]. We obtain good empirical results with the following stochastic graph partitioning technique, termed here *N-run stochastic sampling.*

3.3.1 N-Run Stochastic Sampling

We define a distribution over all edges in G_0, $P(w_i) \propto e^{-\frac{w_i}{T}}$ where T acts as temperature. At each iteration, we draw an edge from this distribution and merge the two vertices. Edge weights to the new vertex formulated by the merge are set to the average of its constituents and the distribution over the edges is recalculated. Merging stops when no positive edges remain in the graph. This procedure is then repeated $r = 1...N$ times and the partitioning with the maximum $\mathcal{F}$ is then selected.

3.4 Coreference Leveraging the Web

Now, consider that we have the ability to augment the graph with additional information using two alternative methods: (1) changing the weight on an existing edge, (2) adding a new vertex and edges connecting it to existing vertices. This new information can be obtained by querying some external source, such as a database or the web.

The first method may be accomplished in author coreference, for example, by querying a web search engine as follows. Clean and concatenate the titles of the citations, issue this query and examine attributes of the returned hits. In this case, a hit indicates the presence of a document on the web that mentions both these titles and hence, some evidence that they are by the same author. Let f_g be this new boolean feature. This feature is then added to an augmented classifier that is then used to determine edge weights.

In the second method, a new vertex can be obtained by querying the web in a similar fashion, but creating a new vertex by using one of the returned web pages as a new mention. Various features $f(\cdot)$ will measure compatibility between the other "citation mentions" and the new "web mention," and with similarly estimated parameters λ, edge weights to the rest of the graph can be set.

In this case, we expand the graph G_0, by adding a new set of vertices, V_1 and the corresponding new set of edges, E_1 to create a new, fully connected graph, G'. Although we are not interested in partitioning V_1, we hypothesize that partitioning G' would improve the optimization of $\mathcal{F}$ on G_0. This can be explained as follows. Let $v_1, v_2 \epsilon V_0$, $v_3 \epsilon V1$, and the edge $< v_1, v_2 >$ has an incorrect, but high negative edge weight. However, the edges $< v_1, v_3 >$ and $< v_2, v_3 >$ have high positive edge weights. Then, by transitivity, partitioning the graph G' will force v_1 and v_2 to be in the same subgraph and improve the optimization of $\mathcal{F}$ on G_0.

(A)..., H. Wang, ... Background Initialization..., ICCV,...2005.
(B)..., H. Wang, ... Tracking and Segmenting People..., ICIP, 2005.
(C)..., H. Wang, ... Gaussian Background Modeling..., ICASSP, 2005.
(D)..., H. Wang, ... Facial Expression Decomposition..., ICCV, 2003.
(E)..., H. Wang, ... Tensor Approximation..., SIGGRAPH. 2005.
(F)..., H. Wang, ... High Speed Machining..., ASME, (JMSE), 2005.

Figure 3.1. Six Example References

As an example, consider the references shown in Fig.3.1. Let us assume that based on the evidence present in the citations, we are fairly certain that the citations A, B and C are by H. Wang 1 and that the citations E and F are by H. Wang 2. Let us say we now need to determine the authorship of citation D. We now add a set of additional mentions from the web, {1, 2, .. 10}. The adjacency matrix of this expanded graph is shown in Fig.3.2. The darkness of the circle represents the level of affinity between two mentions. Let us assume that the web mention 1 (e.g. the web page of H. Wang 1) is found to have strong affinity to the mentions D, E and F. Therefore, by transitivity, we can conclude that mention D belongs to the group 2. Similarly, values in the lower right region could also help disambiguate the mentions through double transitivity.

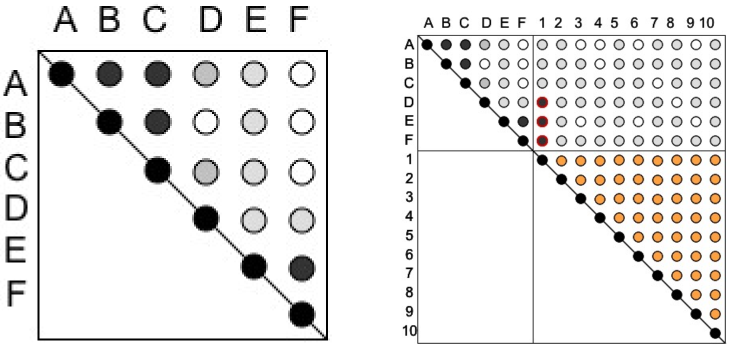

A B C D E F
A B C D E F
A B C D E F 1 2 3 4 5 6 7 8 9 10
A B C D E F 1 2 3 4 5 6 7 8 9 10

Figure 3.2. Extending a pairwise similarity matrix with additional web mentions. A..F are citations and 1..10 are web mentions.

3.5 Resource-bounded Web Usage

We now consider the scenario in which we have a limitation on the resources required to issue queries for all the edges in the fully connected coreference graph and process all the documents obtained as a result of these queries. The two cases to consider are selecting a subset of queries to issue and selecting a subset of nodes to add to the graph.

3.5.1 Selecting a Subset of Queries

Under the constraint on resources, we must select only a subset of edges in E_0, for which we can obtain the corresponding piece of information i_i. Let $E_s \subset E_0$, be this set and I_s be the subset of information obtained that corresponds to each of the elements in E_s. The size of E_s is determined by the amount of resources available. Our objective is to find the subset E_s that will optimize the function $\mathcal{F}$ on graph G_0 after obtaining I_s and applying graph partitioning.

Similarly, in the case of expanded graph G', given the constraint on resources, we must select $V'_s \subset V_1$, to add to the graph. Note that in the context of information gathering from the web, $|V_1|$ is in the billions. Even in the case when $|V_1|$ is much smaller, we may choose to calculate the edge weights for only a subset of E_1. Let $E'_s \subset E_1$ be this set. The sizes of V'_s and E'_s are determined by the amount of resources available. Our objective is to find the subsets V'_s and E'_s that will optimize the function $\mathcal{F}$ on graph G_0 by applying graph partitioning on the expanded graph. We now present the procedure for the selection of E_s.

3.5.1.1 Centroid Based Resource-bounded Information Gathering

For each cluster of vertices that have been assigned the same label under a given partitioning, we define the centroid as the vertex v_c with the largest sum of weights to other members in its cluster. Denote the subset of vertex centroids obtained from clusters as V_c. We can also optionally pick multiple centroids from each cluster. We begin with graph G_0 obtained from the base features of the classifier. We use the following criteria for finding the best order of issuing queries: expected entropy, gravitational force, uncertainty-based and random. Random criteria selects one of the candidate edge randomly at each step. The uncertainty criteria selects an edge based on the entropy of the binary classifier. For each of these criteria, we follow the procedure described below:

1. Partition graph G_0 using N-run stochastic sampling.
2. From the highest scoring partitioned graph G_i^*, find the subset of vertex centroids V_c
3. Construct E_s as the set of all edges connecting centroids in V_c.
4. Order edges E_s into index list I based on the criteria.
5. Using index list I, for each edge $e_i \subset E_s$
    (a) Execute the web query and evaluate additional features from result
    (b) Evaluate classifier for edge e_i with the additional features and form graph G_i from graph G_{i-1}
    (c) Using graph G_i, perform N-run stochastic sampling and compute performance measures

3.5.1.2 Expected Entropy Criterion

1. Force merge of the vertex pair of e_i to get a graph G_p
2. Peform N-run stochastic sampling on G_p. This gives the probabilities p_i for each of the edges in G_p

3. Calculate the entropy, H_p of the graph G_p as follows:

   $H_p = -\sum_i P_i \log P_i$

4. Force split of the vertex pair of e_i to get a graph G_n

5. Repeat steps 2-3 to calculate entropy, H_n for graph G_n

6. The expected entropy, H_i for the edge e_i is calculated as: $H_i = \frac{(H_p)+(H_n)}{2}$ (Assuming equal probabilities for both outcomes)

3.5.1.3 Gravitational Force Criterion

This selection criteria is inspired by the inverse squared law of the gravitational force between two bodies. It is defined as $F = \Gamma\frac{M_1 * M_2}{d^2}$, where Γ is a constant, M_1 and M_2 are analogous to masses of two bodies and d is the distance between them. This criteria ranks highly partitions that are near each other and large, and thus high-impact candidates for merging. Let v_j and v_k be the two vertices connected by e_i. Let C_j and C_k be their corresponding clusters. We calculate the value of F as described above, where M_1 and M_2 are the number of vertices in C_j and C_k respectively. We define $d = \frac{1}{x^{w_i}}$, where w_i is the weight on the edge e_i and x is a parameter that we tune for our method.

3.5.2 Selecting Nodes : RBIG as Set-cover

Incorporating additional nodes in the graph can be expensive. There can be some features between a citation and a web mention (c2w) with high computational cost. Furthermore, the running time of most graph partitioning algorithms depend on the number of nodes in the graph. Hence, instead of adding all the web mentions gathered by pairwise queries, computing the corresponding edge weights and partitioning the resulting graph, it is desirable to find a minimal subset of the web documents that would help bring most of the coreferent citations together. This is equivalent to selectively filling the entries of the upper right section of the matrix. We observe that this problem is similar to the classic Set-cover problem with some differences as noted below.

The standard Set-cover problem is defined as follows. Given a finite set U and a collection $C = \{S_1, S_2, ..., S_m\}$ of subsets of U. Find a minimum sized cover $C' \subseteq C$ such that every element of U is contained in at least one element of C'. It is known that greedy approach provides an $\Omega(\text{ln n})$ approximation to this NP-Complete problem.

We now cast the problem of *Resource-bounded information gathering* using additional web mentions as a variant of Set-cover. The goal is to "cover" all the citations using the least possible number of web pages, where "cover" is loosely defined by some heuristic. Assuming a simplistic, "pure" model of the web (i.e. each web page "covers" citations of only one author), we can think of each web page as a set of citations and the set of citations by each author as the set of elements to be covered. We now need to choose a minimal set of web pages such that they can provide information about most of the citations in the data.

There are some differences between Set-cover and our problem that reflect the real life scenario as follows. There can be some elements in U which are not covered by any elements in C. That is, $\bigcup S_i \neq U$. Also, in order for the additional web page to be useful for improving coreference accuracy in the absence of a strong w2w classifier, it has to cover at least two elements. Keeping these conditions in mind, we modify the greedy solution to Set-cover as shown in Algorithm 1.

3.5.3 Selecting Queries: Inter-cluster and Intra-cluster queries

In many scenarios, issuing queries and obtaining the results is itself an expensive task. In our previous methods, we used all possible pairwise queries to obtain additional web documents. In this section, we will use the information available in the

Algorithm 1 RBIG-Set-cover Algorithm

```
Input:
Set of citations $U$
Collection of web documents $C : \{S_1, S_2, ..., S_n\}$
$O \Leftarrow \emptyset$
while $U$ is “coverable” by $C$ do
    $S_k \Leftarrow arg\max_{S_i \in C} |S_i|$
    $O \Leftarrow O \cup \{S_k\}$
    $U \Leftarrow U \cap S_k$
    $C \Leftarrow \{S_i | S_i = S_i \cap S_k\}$
end while
return $O$
$U$ is “coverable” by $C \equiv \exists_{(e \in U \wedge S_i \in C)}(e \in S_i)$
```

test data (upper left section of the matrix) to selectively issue queries, such that the results of those queries would have most impact on the accuracy of coreference.

The first method for reducing the number of web queries is to query only a subset of the edges between current partitions. We start by running the citation-to-citation classifier on the test data and obtain some initial partitioning. For each cluster of vertices that have been assigned the same label under a given partitioning, we define the centroid as the vertex with the largest sum of weights to other members in its cluster. We connect all the centroids with each other and get a collection of queries, which are then used for querying the web. Let n be the number of citations in the data and m be the number of currently predicted authors. Assuming that the baseline features provide some coreference information, we have reduced the number of queries to be executed from $O(n^2)$ to $O(m^2)$. A variation of this method picks multiple centroids, proportional to the size of each initial partition, where the proportion can be dictated by the amount of resources available.

The second method for reducing the number of web queries is to query only a subset of the edges within current partitions. As before, we first start by running the citation-to-citation classifier on the test data and then obtain some initial partitioning. For each initial partition, we select two most tightly connected citations to form a query. Under the same assumptions stated above, we have now reduced the number of queries to be executed from $O(n^2)$ to $O(m)$. A variation of this method picks more than two citations in each partition, including some random picks.

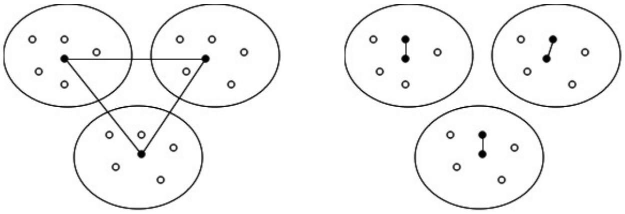


Figure 3.3. Inter-cluster and Intra-cluster queries

Both these approaches are useful in different ways. Inter-cluster queries help find evidence that two clusters should be merged, whereas intra-cluster queries help find additional information about a hypothesized entity. The efficiency of these two methods depend on the number of underlying real entities as well as the quality of initial partitioning.

3.5.4 Hybrid Approach

For large scale system, we can imagine combining the two approaches, i.e. *Selecting Nodes* and *Selecting Queries* to form a hybrid approach. For example, we can first select queries using, say intra-cluster queries to obtain additional mentions. This would help reduce querying cost. We can then reduce the computation cost by selecting a subset of the web mentions using the Set-cover method. We show experimentally in the next section that this can lead to a very effective strategy.

3.5.5 Cost-Benefit Analysis

It should be noted that the choice of strategy for *Resource-bounded information gathering* in the case of expanded graph should be governed by a careful Cost-Benefit analysis of various parameters of the system. For example, if the cost of computing correct edge weights using fancy features on the additional mentions is high, or if we are employing a graph partitioning technique that is heavily dependent on the number of nodes in the graph, then the Set-cover method described above would be effective in reducing the cost. On the other hand, if the cost of making a query and obtaining additional nodes is high, then using inter-cluster or intra-cluster methods is more desirable. For a large scale system, a hybrid of these methods could be more suitable.

3.6 Experimental Results

3.6.1 Dataset and Infrastructure

We use the Google API for searching the web. The data sets used for these experiments are a collection of hand labeled citations from the DBLP and Rexa corpora (see table 3.1). The portion of DBLP data, which is labeled at Pennstate University is referred to as ‘Penn’. Each dataset refers to the citations authored by people with the same last name and first initial. The hand labeling process involved carefully segregating these into subsets where each subset represents papers written by a single real author.

The ‘Rbig’ corpus consists of a collection of web documents which is created as follows. For every dataset in the DBLP corpus, we generate a pair of titles and issue queries to Google. Then, we save the top five results and label them to correspond with the authors in the original corpus. The number of pairs in this case corresponds to the sum of the products of the number of web documents and citations in each dataset.

All the corpora are split into training and test sets roughly based on the total number of citations in the datasets. We keep the individual datasets intact because it would not be possible to test graph partitioning performance on randomly split citation pairs.

Corpus	# Sets	# Authors	# Citations	# Pairs
DBLP	18	103	945	43338
Rexa	8	289	1459	207379
Penn	7	139	2021	455155
Rbig	18	103	1360	126205

Table 3.1. Summary of Data set properties.

3.6.2 Baseline, Graph Partitioning, and Web Information as a Feature

The maximum entropy classifier for calculating the edge weights is built using the following features. We use the first and middle names of the author in question and the number of overlapping co-authors. The US census data helps us determine how rare the last name of the author is. We use several different similarity measures on the titles of the two citations, such as, the cosine similarity between the words, string edit distance, TF-IDF measure and the number of overlapping bigrams and trigrams. We also look for similarity in author emails, institution affiliation and the venue of publication if available. We use a greedy agglomerative graph partitioner in this set of experiments.

The baseline column in table 3.4 shows the performance of this classifier. Note that there is a large number of negative examples in this dataset and hence we prefer pairwise F1 over accuracy as the main evaluation metric. Table 2 shows that graph partitioning significantly improves pairwise F1. We also use area under the ROC curve for comparing the performance of the pairwise classifier, with and without the web feature.

Note that these are some of the best results in author coreference and hence qualify as a good baseline for our experiments with the use of web. It is difficult to make direct comparison with other coreference schemes [36] due to the difference in the evaluation metrics.

Table 3.4 compares the performance of our model in the absence and in the presence of the Google title feature. As described before, these are two completely identi-

Method		AROC	Acc	Pr	Rec	F1
Baseline	class.	.847	.770	.926	.524	.669
DBLP	part.	-	.780	.814	.683	.743
W/ Google	class.	.913	.883	.907	.821	.862
DBLP	part.	-	.905	.949	.830	.886
Baseline	class.	.866	.837	.732	.651	.689
Rexa	part.	-	.829	.634	.913	.748
W/ Google	class.	.910	.865	.751	.768	.759
Rexa	part.	-	.877	.701	.972	.814
Baseline	class.	.688	.838	.980	.179	.303
Penn	part.	-	.837	.835	.211	.337
W/ Google	class.	.880	.913	.855	.672	.752
Penn	part.	-	.918	.945	.617	.747

Figure 3.4. Effect of using the Google feature. Top row in each corpus indicates results for pairwise classification and bottom row indicates results after graph partitioning.

cal models, with the difference of just one feature. The F1 values improve significantly after adding this feature and applying graph partitioning.

3.6.3 Expanding the Graph by Adding Web Mentions

In this case, we augment the citation graph by adding documents obtained from the web. We build three different kinds of pairwise classifiers to fill the entries of the matrix shown in fig. 3.2. The first classifier, between two citations, is the same as the one described in the previous section. The second classifier, between a citation and a web mention, predicts whether they both refer to the same real author. The features for this second classifier include: occurrence of the citation's author and coauthor names, title words, bigrams and trigrams in the web page. The third classifier, between two web mentions, predicts if they both refer to the same real author or not. Due to the sparsity of training data available at this time, we set the value of zero in this region of the matrix, indicating no preference. We now run the greedy agglomerative graph partitioner on this larger matrix and finally, measure the results on the upper left matrix.

We compare the effects of using web as a feature and web as a mention on the DBLP corpus. We use the Rbig corpus for this experiment. Table 3.2 shows that the use of web as a mention improves the performance on F1. Note that alternative query schemes may yield better results.

Data	Acc.	Pr.	Rec.	F1
Baseline	.7800	.8143	.6825	.7426
Web Feature	.9048	.9494	.8300	.8857
Web Mention	.8816	.8634	.9462	.9029

Table 3.2. DBLP Results when using Web Pages as Extra Mentions

3.6.4 Applying the Resource Bounded Criteria for Selective Querying

We now turn to the experiments that use different criteria for selectively querying the web. We present the results on test datasets from DBLP and Rexa corpora. As described in the previous section, the query candidates are the edges connecting centroids of initial clustering. We use multiple centroids and pick top 20% tightly connected vertices in each cluster. We experiment with ordering these query candidates according to the four criteria: expected entropy, gravitational force, uncertainty-based and random. For each of the queries in the proposed order, we issue a query to Google and incorporate the result into the binary classifier with an additional feature.

If the prediction from this classifier is greater than a threshold (t = 0.5), we force merge the two nodes together. If lower, we have two choices. We can impose the force split, in accordance with the definition of expected entropy. We call this approach "split and merge". The second choice is to not impose the force split, because, in practice, Google is not an oracle and absence of co-occurence of two citations on the web is not an evidence that they refer to different people. We call this approach

Method	Precision	Recall	F1
Merge Only			
Expected Entropy	73.72	87.92	72.37
Gravitational Force	63.10	92.37	64.55
Uncertainty	64.95	87.83	63.54
Random	63.97	89.46	64.23
Merge and Split			
Expected Entropy	76.19	58.56	60.90
Gravitational Force	64.10	53.06	53.56
Uncertainty	66.56	54.45	55.32
Random	66.45	50.47	52.27
No Merge			
Expected Entropy	91.46	38.46	51.06
Gravitational Force	91.53	37.84	50.47
Uncertainty	87.01	41.91	52.70
Random	86.96	43.77	54.03

Table 3.3. Area Under Curve for different Resource Bounded Information Gathering criteria

"merge only". The third choice, is to simply incorporate the result of the query into the edge weight.

After each query, we rerun the stochastic partitioner and note the precision, recall and F1. This gives us a plot for a single dataset. Note that the number of proposed queries in each dataset is different. We get an average plot by sampling the result of each of the datasets for a fixed number of points, n (n = 100). We interpolate when queries fewer than n are proposed. We then average across these datsets and calculate the area under these curves, as shown in table 3.3.

These curves measure the effectiveness of a criteria in achieving maximum possible benefit with least effort. Hence, a curve that rises the fastest, and has the maximum area under the curve is most desired. Expected entropy approach, gives the best performance on F1 measure, as expected.

It is interesting to note that the gravitational-force-based criteria does better than the expected entropy criteria on recall, but worse on the precision. We hypothesize that this is because gravitational approach captures the sizes of the two clusters and hence tends to merge large clusters, without paying much attention to the 'purity' of the resulting clusters. The expected entropy approach, on the other hand, takes this into account and hence emerges as the best method. In the future, we would like to verify this hypothesis experimentally. The force-based approach is a much faster approach and it can be used as a heuristic for very large datasets.

Both the criteria work better than uncertainty-based and random, except an occasional spike. All four methods are sensitive to the noise in data labeling, result of the web queries and sampling in stochastic graph partitioning, as reflected by the spikes in the curves. However, these results show that expected entropy approach is the best way to achieve maximum returns on investment and proves to be a promising approach to solve this class of problems, in general.

3.6.5 Resource Bounded Querying for Additional Web Mentions: Intra-Setcover Hybrid Approach

Finally, we present the results of the hybrid approach on the DBLP corpus. In Fig.3.5, the black series plots the ratio of the number of documents added to the graph in each method to the number of documents obtained by all pairwise queries. This represents cost. The gray series plots the ratio of the improvement obtained by each method to the maximum achievable improvement (using all mentions and queries). This represents benefit. For the Intra-Setcover hybrid approach, we achieve 74.3% of the total improvement using only 18.3% of all additional mentions.

3.7 Open Theoretical Problem

The problem of Resource-bounded information gathering for entity resolution extends to a much larger class of interesting problems. We propose this as an open theoretical problem.

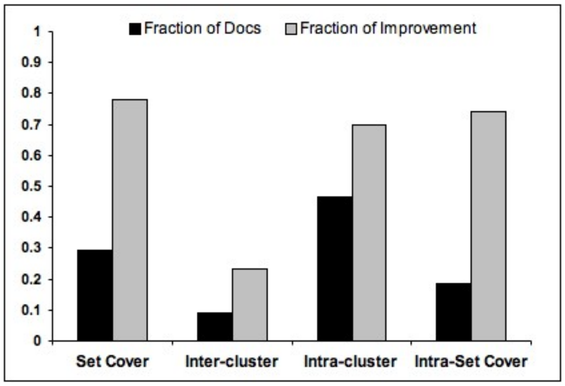
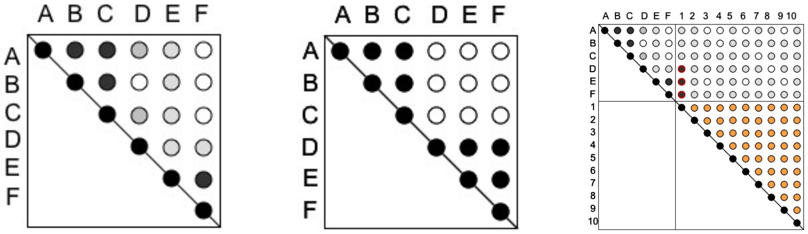


Figure 3.5. DBLP: For each method, fraction of the documents obtained using all pairwise queries and fraction of the possible performance improvement obtained. Intra-Setcover hybrid approach yields the best cost-benefit ratio

The standard correlation clustering problem on a graph with real-valued edge weights is as follows: there exists a fully connected graph $G(V, E)$ with n nodes and edge weights, $w_{ij} \in [-1, +1]$. The goal is to partition the vertices in V by minimizing the inconsistencies with the edge weights [5]. That is, we want to find a partitioning that maximizes the objective function $\mathcal{F} = \sum_{ij} w_{ij} f(i, j)$, where $f(i, j) = 1$ when v_i and v_j are in the same partition and -1 otherwise.

Now consider a case in which there exists some "true" partitioning $\mathcal{P}$, and the edge weights $w_{ij} \in [-\infty, +\infty]$ are drawn from a random distribution (noise model) that is correlated with whether or not edge $e_{ij} \in E$ is cut by a partition boundary. The goal is to find an approximate partitioning, $\mathcal{P}_a$, of V into an unknown number of k partitions, such that $\mathcal{P}_a$ is as 'close' to $\mathcal{P}$ as possible. There are many different possible measures of closeness to choose from. Let $\mathcal{L}(\mathcal{P}, \mathcal{P}_a)$ be some arbitrary loss function. If no additional information is available, then we could simply find a partitioning that optimizes $\mathcal{F}$ on the given weights.

We consider settings in which we may issue queries for additional information to help us reduce loss $\mathcal{L}$. Let $G_0(V_0, E_0)$ be the original graph. Let $\mathcal{F}_0$ be the objective function defined over G_0. Our goal is to perform correlation clustering and optimize


A B C D E F
A B C D E F
A B C D E F
A B C D E F
A B C D E F 1 2 3 4 5 6 7 8 9 10
A B C D E F 1 2 3 4 5 6 7 8 9 10

(a) G_0 (b) Result of Q1 (c) Result of Q2

Figure 3.6. Results of the two kinds of queries. (a) The adjacency matrix of G_0 where darker circles represent edges with higher weight. (b) The new edge weights w'_{ij} after issuing the queries from Q1. (c) The graph expanded after issuing queries from Q2. The upper left corner of the matrix corresponds to G_0 and the remaining rows and columns correspond to the nodes in V_1.

$\mathcal{F}_0$ with respect to the true partitioning of G_0. We can augment the graph with additional information using two alternative methods: (1) updating the weight on an existing edge, (2) adding a new vertex and edges connecting it to existing vertices. We can obtain this additional information by querying a (possibly adversarial) oracle using two different types of queries. In the first method, we use query of type Q1, which takes as input edge e_{ij} and returns a new edge weight w'_{ij}, where w'_{ij} is drawn from a different distribution that has higher correlation with the true partitioning $\mathcal{P}$.

In the second method, we can expand the graph G_0, by adding a new set of vertices, V_1 and the corresponding new set of edges, E_1 to create a larger, fully connected graph, G'. Although we are not interested in partitioning V_1, we hypothesize that partitioning G' would improve the optimization of $\mathcal{F}\prime$ on G_0 due to transitivity of partition membership. In this case, given resource constraints, we must select $V'_s \subset V_1$ to add to the graph. These can be obtained by second type of query, Q2, which takes as input (V_0, E_0) and returns a subset $V'_s \subset V_1$. Note that the additional nodes obtained as a result of the queries of type Q2 help by inducing a new, and presumably more accurate partitioning on the nodes of G_0. Fig. 3.6 illustrates the result of these queries. However, there exist many possible queries of type Q1 and Q2, each with an associated cost. There is also a cost for performing computation on the additional information. Hence, we need an efficient way to select and order queries under the given resource constraints.

Formally, we define the problem of *resource-bounded information gathering for correlation clustering* as follows. Let $c(q)$ be the cost associated with a query $q \in Q1 \cup Q2$. Let b be the total budget on queries and computation. Find distinct queries $q_1, q_2, ...q_m \in Q1 \cup Q2$ and $\mathcal{P}_a$, to minimize $\mathcal{L}(\mathcal{P}, \mathcal{P}_a)$, s.t. $\sum_{q_i} c(q_i) \leq b$.

3.8 Chapter Summary

In this chapter, we learn that when acquiring information for a structured problem (in this case, a graph partitioning one), it is preferable to reduce uncertainty in the overall structure (graph), rather than focusing on reducing only local uncertainty. In our example, we demonstrate that we can allocate resources more effectively, by selecting an edge, such that improving the corresponding edge weight reduces the expected entropy of the entire graph. In the future, it would be interesting to develop a query selection criteria that adapts its decision based on the changes after acquiring information, rather than ranking all the queries initially. We also show that additional information can be incorporated in the form of additional nodes in the graph, which can aid more accurate partitioning; an idea that can potentially be applied in many interesting, real world domains.

To the best of my knowledge, our work is the first to propose acquisition of external information for improving an entity resolution problem that is cast as a graph partitioning problem, and demonstrating how to do it efficiently under limited resources. We believe that this problem setting has the potential to bring together ideas from the areas of active information acquisition, relational learning, decision theory and graph theory, and apply them in real world domains. This work also leads to interesting theoretical questions, whose answers can expand our understanding of how external information can be used efficiently to improve clustering problems. Some interesting directions for this work are : analytically quantifying the effect of changing a single edge weight on the partitioning of the entire graph; estimating the probability of recovering the true partition under various query selection strategies for general random graphs and possible directions for approximations; and general techniques for selectively acquiring information for expanding graphs.

CHAPTER 4

PREDICTION-TIME ACTIVE FEATURE-VALUE ACQUISITION FOR CUSTOMER TARGETING

4.1 Introduction

In the previous chapter, we selected a subset of instances for which to obtain a single feature value. We now focus on acquiring multiple feature values from external sources such as the web or an information vendor. The previous chapter assumes that the input instances are interdependent, which directly affects the criterion for selecting *query* actions. In this chapter, we will develop the ideas for query selection for the case of i.i.d. input instances. We do not focus on the *download* and *extract* actions from the RBIA framework in Chapter 1, and assume that the information from external source is available in processed form, after a *query* action is performed. The *db-inference* in this case involves predicting the value of a target variable using all available information. Once again, the *db-inference* impacts our methods for selecting most effective *query* actions.

The cost-effective acquisition of data for modeling and prediction has been an emerging area of study which, in the most general case, is referred to as Active Information Acquisition [77]. We examine the specific case of this problem, where a set of features maybe missing and all missing feature values can be acquired for a selected instance[62]. This Instance-completion setting allows for computationally cheap yet very effective heuristic approaches to feature-acquisition. It has been shown that at train time, actively selecting feature values to acquire results in building effective models at a lower cost than randomly acquiring features [63]. We now study prediction-time Active Feature-value Acquisition (AFA) in the context of different customer targeting domains.

Our first domain is a system developed at IBM to help identify potential customers and business partners. The system formerly used only structured firmographic data to predict the propensity of a company to buy a product. Recently, it has been shown that incorporating information from company websites can significantly improve these targeting models. However, in practice, processing websites for millions of companies is not desirable due to the processing costs and noisy web data. Hence we would like to select only a subset of companies for which to acquire web-content, to add to the firmographic data, to aid in prediction. This is a case of the Instance-completion setting, in which firmographic features are available for all instances, and the web features are missing and can be acquired at a cost. Instance-completion heuristics have been applied to this data during induction [61]; and, here, we study the complementary task of prediction-time AFA. An interesting aspect observed in [61] is that web content can also be noisy, and active-selection of web-content can often do better than using all web-content. This shows that prediction-time AFA can also be used in the context of data cleaning problems.

The second domain is a web-usage study by Zheng and Padmanabhan [98]. Their data set contains information about web users and their visits to retail web sites. The given features describe a visitor's surfing behaviors at a particular site, and the additional features, which can be purchased at a cost from an external vendor, provides aggregated information about the same visitor's surfing behavior on other e-commerce sites. The target variable indicates whether or not the user made a purchase during a given session. This setting also fits naturally in the Instance-completion setting of AFA.

These domains exhibit a natural dichotomy of features, in which one set of features is available for all instances, and the remaining features can be acquired, as a set, for selected instances. As such, these domains lend themselves to AFA in the Instance-completion setting, and have been used in the past in studies of feature-acquisition during induction [62]. At the time of induction, class labels are available for all instances — including the incomplete instances. This information can be used effectively to estimate the potential value of acquiring more information for the incomplete instances. However, this label information is obviously not present during prediction on test instances, and as such leads us to explore alternative acquisition strategies. In particular, we explore methods to estimate the expected benefit of acquiring additional features for an incomplete instance, versus making a prediction using only incomplete feature information. Extensive experimental results confirm that our approaches can effectively select instances for which it is beneficial to acquire more information to classify them better, as compared to acquiring additional information for the same number of randomly sampled instances.

4.2 General Problem Setup

Assume that we are given a classifier induced from a training set consisting of n features and the class labels. We are also given a test set of m instances, where each instance is represented with n feature values. This test set can be represented by the matrix F, where $F_{i,j}$ corresponds to the value of the j^{th} feature of the i^{th} instance. The matrix F may initially be incomplete, i.e., it contains missing values. At prediction time, we may acquire the value of $F_{i,j}$ at the cost $C_{i,j}$. We use $q_{i,j}$ to refer to the query for the value of $F_{i,j}$. The general task of prediction-time AFA is the selection of these instance-feature queries that will result in the most accurate prediction over the entire test set at the lowest cost.

4.3 Prediction-time Active Feature-value Acquisition for Instance-completion

As noted earlier, the generalized AFA setting has been studied previously for induction-time. Under the induction-time AFA setting, the training instances have missing features values, which can be acquired at a cost and the goal is to learn the most accurate model with the lowest cost. This model is usually tested on a test-set of complete instances. Here, we are interested in the complementary task of Active Feature-value Acquisition at the time of prediction. The fundamental difference between these two settings is that for induction-time AFA, our goal is to learn a model that would make most accurate predictions on a test set with complete instances, whereas, for prediction-time AFA, the model is trained from a set of complete instances, and the goal is to select queries that will lead to most accurate prediction on incomplete test instances. A third scenario is when the feature values are missing at both induction and prediction time, and the learner is aware of the cost constraints at prediction-time. Hence, the goal of the learner is to learn the most accurate model that optimizes cost at both train and test time. In future, we would like to explore this third scenario.

Here, we consider a special case of the prediction-time AFA problem mentioned above; where feature values for an instance may naturally be available in two sets — one set of features is given for all instances, and the second set can be acquired from an external source at a cost. The task is to select a subset of instances for which the additional features should be acquired to achieve the best cost-benefit ratio.

The two sets of features can be combined in several ways to build a model (or make a prediction at test time). The features from the two sets can be merged before building a model, which is referred to as *early fusion*. Alternatively, two separate models are built using the two sets of features and their outputs are combined in some way to make the final prediction — known as *late fusion*. The alternative strategy we employ in our work is called *Nesting* [61] — in which we incorporate the output of a model using the second set of *additional* features (inner model) as an input to the model using the first set of *given* features (outer model). Specifically, we add another feature in the outer model, corresponding to the predicted probability score for the target variable, as given by the inner model.

The general framework for performing prediction-time AFA for instance-completion setting is described in Algorithm 1. We assume that we are given two models, one induced only from the given features and another one induced from both given and additional features. At prediction time, we are given a set of incomplete instances. We compute a score for each of the incomplete instances based on some acquisition strategy. We sort all instances based on this score and acquire additional features in the sorted order until some stopping criterion is met. The final prediction is made using the appropriate model on the entire set of instances. Note that induction-time AFA has a similar framework, but the main difference is that at induction-time, after each batch of feature acquisition, we need to relearn the model, and hence, recompute the score. On the other hand, at prediction-time, acquiring additional features for one instance has no effect on the prediction of another instance, and as such we can generate the score on the entire set once before starting the acquisition process. This makes large scale, prediction-time AFA feasible on a variety of domains. Note that if the prediction algorithm takes into account the values of multiple test instances, our method can not be directly applied. In the next section we describe alternative approaches to selecting instances for which to acquire additional feature values.

4.4 Acquisition Strategies

4.4.1 Uncertainty Sampling

The first AFA policy we explore is based on the uncertainty principle that has been extensively applied in the traditional active learning literature [49], as well as

Algorithm 2 Prediction-time AFA for Instance-completion using Nesting

```
Given:
$I$ - Set of incomplete instances, which contain only given features
$C$ - Set of complete instances, which contain both given and additional features
$T$ - Set of instances for prediction, $I \wedge C$
$M_g$ - Model induced from only given features
$M_c$ - Model induced from both given and additional features
$\forall x_j \in I$, compute the score $S = Score(M_g, x_j)$, based on the AFA strategy
Sort instances in $I$ by score, S.
Repeat until stopping criterion is met
    Let $x_j$ be the instance in $I$ with the next highest score
    Model $M = M_g$ if $x_j \in I$ and $M = M_c$ if $x_j \in C$
return Predictions on $T$ using the appropriate model M
```

previous work on AFA [62]. In Uncertainty Sampling we acquire more information for a test instance if the current model cannot make a confident prediction of its class membership. There are different ways in which one could measure uncertainty. In our study, we use unlabeled margins [62] as our measure; which gives us the same ranking of instances as entropy, in the case of binary classification. The unlabeled margin captures the model's ability to distinguish between instances of different classes. For a probabilistic model, the absence of discriminative patterns in the data results in the model assigning similar likelihoods for class membership of different classes. Hence, the Uncertainty score is calculated as the absolute difference between the estimated class probabilities of the two most likely classes. Formally, for an instance x, let $P_y(x)$ be the estimated probability that x belongs to class y as predicted by the model. Then the Uncertainty score is given by $P_{y1}(x) - P_{y2}(x)$, where $P_{y1}(x)$ and $P_{y2}(x)$ are the first-highest and second-highest predicted probability estimates respectively. Here, a lower score for an instance corresponds to a higher expected benefit of acquiring additional features.

4.4.2 Expected Utility

Uncertainty Sampling, as described above, is a heuristic approach that prefers acquiring additional information for instances that are currently not possible to classify with certainty. However, it is possible that additional information may still not reduce the uncertainty of the selected instance. The decision theoretic alternative is to measure the expected reduction in uncertainty for all possible outcomes of a potential acquisition. According to an optimal strategy, the next best instance, for which we should acquire features is the one that will result in the greatest reduction in uncertainty per unit cost, in expectation. Since true values of missing features are unknown prior to acquisition, it is necessary to estimate the potential impact of every acquisition for all possible outcomes. Ideally, this requires exhaustively evaluating all possible combinations of values that the additional (missing) features can take for each instance. However, in our Nesting approach to combining feature sets, we reduce the additional features into a single score, which is used as a feature along with the other given features. This allows us to dramatically simplify the complexity of this approach, by only treating this score as a single missing feature, and estimating the utility of possible values it can take. Of course, calculating expectation over this single score does not give us the true utility of the additional features, but it makes the utility computation feasible, especially when we have a very large number of additional features. As such, the expected utility can be computed as:

$$EU(q_j) = \int_x U(S_j = x, C_j) P(S_j = x) \tag{4.1}$$

Where, $P(S_j = x)$ is the probability that S_j has the value x and $U(S_j = x, C_j)$ is the utility of knowing that S_j has value x. In other words, it is the benefit arising from obtaining a specific value x for score S_j, at cost C_j. In practice, in order to compute the expected utility, we discretize the values of S and replace the integration in Eq. 4.1 with piece-wise summation. The two terms, U and P in Eq. 4.1 must be estimated only from available data. We discuss how we empirically estimate these quantities below.

Estimating utility: The utility measure, U, can be defined in one of several different ways. In the absence of class labels, we resort to using measures of uncertainty of the model prediction as a proxy for prediction accuracy. One obvious choice here is to measure the reduction in entropy of the classifier after obtaining value x — similar to what is done in traditional active learning [75], i.e.,

$$U(S_j = x, C_j) = -\frac{H(X \wedge S_j = x) - H(X)}{C_j} \tag{4.2}$$

Where, $H(X \wedge S_j = x)$ is the entropy of the classifier on the instance with features X, augmented with $S_j = x$, $H(X)$ is the entropy of the classifier on the instance with features X and C_j is the cost of feature score S_j.

However, using reduction in entropy may not be ideal. We illustrate this through Fig. 4.1, which compares entropy and unlabeled margins as a function of the predicted class membership probability, $\hat{p}(y|x)$. Note that it does not matter which class y we choose here. We see from the figure, that for the same Δx difference in class membership probability, the corresponding reductions in entropy are different. In particular, the further we are from the decision boundary the higher the change in entropy, i.e. $\Delta y_2 > \Delta y_1$. All else being equal, this measure would prefer acquisitions that would reduce entropy further from the classification boundary; which is less likely to affect the resulting classification. Alternatively, one could use unlabeled margins, which is a linear function of the probability estimate on either side of the decision boundary. This gives the following *expected unlabeled margin* utility measure:

$$U(S_j = x, C_j) = \frac{UM(X \wedge S_j = x) - UM(X)}{C_j} \tag{4.3}$$

Where, $UM(X)$ is the unlabeled margin as described in Sec. 4.4.1.

Furthermore, one might choose to prefer a difference in $\hat{p}$ closer to the decision boundary; since this is more likely to result in an alternative classification for an instance. We can capture this relationship, by using the log of the unlabeled margins, which gives us the following *expected log margin* measure of utility:

$$U(S_j = x, C_j) = \frac{\ln(UM(X \wedge S_j = x)) - \ln(UM(X))}{C_j} \tag{4.4}$$

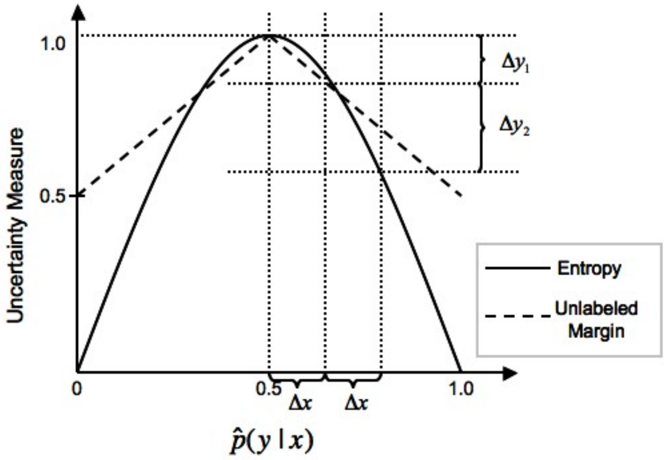


Figure 4.1. Comparison of unlabeled margin and entropy as measures of uncertainty.

Estimating feature-value distributions: Since the true distribution of the score S_j is unknown, we estimate $P(S_j = x)$ in Eq.1 using a probabilistic learner. We start by dropping the class variables from the training instances. Next, we use a model trained only on the additional features to predict the value of S_j and discretize it. We now use S_j as the target variable and all given features as the predictors to learn a classifier M. When evaluating the query q_j, the classifier M is applied to instance X_j to produce the estimate $\hat{P}(S_j = x)$.

4.5 Empirical evaluation

We tested our proposed feature-acquisition approaches on the following data sets. *Rational*, comes from a system developed at IBM to help identify potential customers and business partners. The remaining three data sets come from the web usage study by Zheng and Padmanabhan [98].

Dataset	Model using given features	Composite model
bmg	77.41	88.11
expedia	87.07	94.53
qvc	81.04	88.94

Table 4.1. Improvement in Accuracy after using additional features. The AUC value for Rational dataset, goes from 79.0 to 82.3 after acquiring additional features.

4.5.1 Comparison of acquisition strategies

For all datasets, we use Nesting to combine the two separate feature sets. We experimented with different combinations of base classifiers in Nesting, and found that using decision trees for the additional features and logistic regression for the composite model is most effective for the web-usage datasets. For *Rational*, we use multinomial naive Bayes for the web features, and logistic regression for the composite model. Since there is a small proportion of instances from the target class in *Rational*, and it is a ranking problem, we use AUC instead of accuracy as a performance metric (as done in [61]). For all other datasets, we use accuracy as done in their previous usage [98].

Table 4.1 shows improvement in accuracy of the classification model after acquiring additional features. In all four experiments, the models using the additional features performed statistically significantly better than the models on given features alone, based on paired t-tests ($p < 0.05$).

We ran experiments to compare Random Sampling and the AFA strategies described in Sec. 4.4. The performance of each method was averaged over 10 runs of 10-fold cross-validation. In each fold, we generated acquisition curves as follows. After acquiring additional features for each actively-selected test instance, we measure accuracy (or AUC, in case of *Rational*) on the entire test set using the appropriate model (see Algorithm 1). In the case of Random Sampling, instances are selected uniformly at random from the pool of incomplete instances. For the expected utility approaches described in Sec. 4.4.2, we used 10 equal-width bins for the discretization of the score S_j in Eq. 4.1.

Fig. 4.2 shows the effectiveness of each strategy in ordering the instances so as to get the most benefit with the least cost of data acquisition. We assume, for these experiments, that there is a unit cost of acquiring additional features for each instance. In all cases, active acquisition clearly out-performs Random Sampling, resulting in improved prediction performance for the same amount of feature information acquired for the test instances. Also, a large amount of improvement in accuracy is achieved by acquiring complete feature sets for only a small fraction of instances, which suggests that it is not critical to have complete feature information for all instances to correctly classify them.

In the web usage datasets, unlabeled margin does better than all other measures of uncertainty. Also, note that expected log margin performs slightly better than other utility measures. It is interesting to note that the prediction on one instance is completely independent of the acquisition of additional feature on another instance. This is one reason why unlabeled margin proves to be an effective method for prediction-time AFA in most cases.

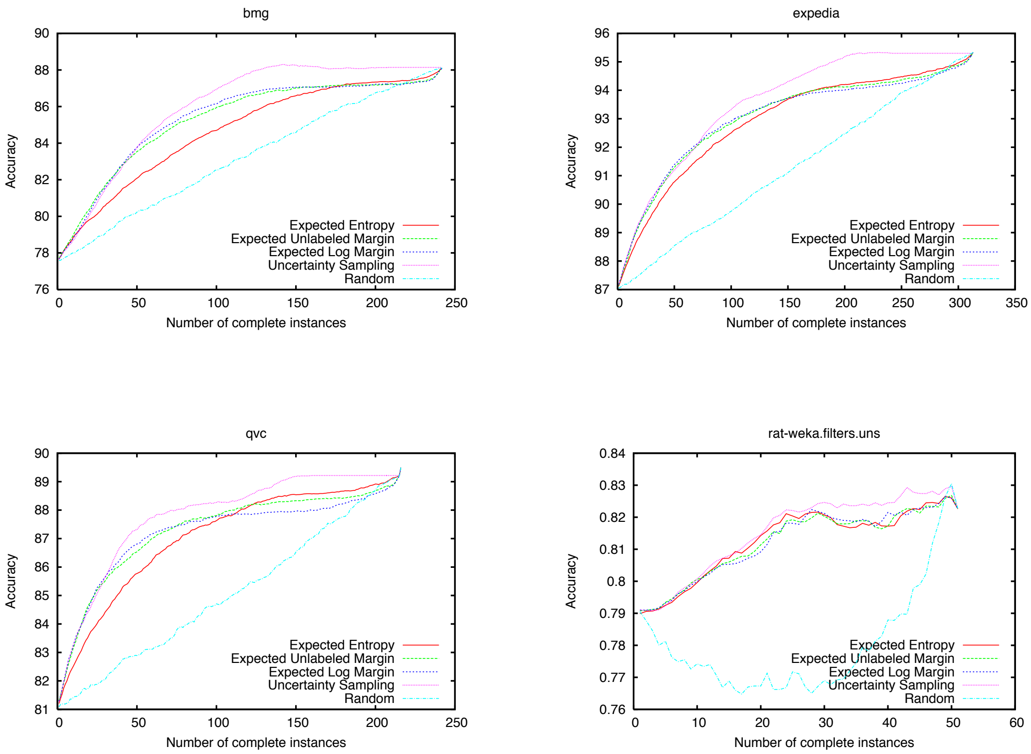


Figure 4.2. Comparison of acquisition strategies

4.5.2 Oracle study and discussion

Even with the gross approximations and estimations done in Sec. 4.4.2, the *Expected Utility* approach still manages to perform quite well compared to random sampling. Furthermore, using reduction in log margins tends to slightly outperform the alternative utility measures, for the reasons discussed in Sec. 4.4.2. However, in general, the *Expected Utility* methods still do not exceed the performance of *Uncertainty Sampling*, as one would expect. It is possible that the estimations done in the computation of *Expected Utility* are too crude and need to be improved. One source of improvement could be through better estimation of the probability distribution of missing feature values. Currently this is being reduced to estimating the probability of a single discretized score, representing the output of a model built using the addi-

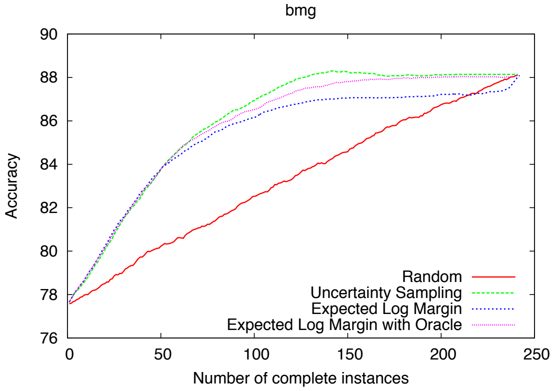


Figure 4.3. Comparison of acquisition strategies using an Oracle

tional features. In order to evaluate the room for improvement in this estimation, we use the true value of the discretized score while calculating the expectation in Eq. 4.1. This *Expected Log Margins* with Oracle approach is shown in Fig. 4.3, in comparison to the estimated *Expected Log Margins* approach. We see that, indeed, if we had the true probability estimate $P(S_j = x)$, we can perform much better than using the estimation approach described in Sec. 4.4.2. However, this by itself is still insufficient to outperform *Uncertainty Sampling*. We may be losing too much information by compressing the additional feature set into a single score. Using alternative feature-reduction techniques may lead to a more meaningful estimation of the missing value distribution, without too much increase in computational complexity brought about by having to estimate the joint distribution of features. Perhaps a better estimate of utility U is also required to make the *Expected Utility* approach more effective.

In summary, we demonstrate that our approaches of measuring the uncertainty of predictions, and the expected reduction of uncertainty through additional feature-acquisition, are much more effective than the baseline approach of uniformly sampling instances for acquiring more information. Empirical results show that estimating the expected reduction in uncertainty of a prediction is an effective acquisition strategy. However, it is not as effective as just selecting instances based on the uncertainty of their prediction using incomplete information.

4.6 Chapter Summary

In this chapter, we apply various instance selection criteria for *query* actions that help acquire additional features at test time for a classification problem. We can select an instance that is most uncertain, as predicted by existing features, such that new features would lead to a more certain classification. Alternatively, we can select an instance for which the new features will reduce uncertainty *in expectation*. We show how to address the problem of unavailability of class labels at test time for computing the value of obtaining additional information for an incomplete instance and study the effectiveness of these methods on customer targeting applications.

CHAPTER 5

RESOURCE-BOUNDED INFORMATION EXTRACTION USING INFORMATION PROPAGATION

5.1 Introduction

The goal of traditional information extraction is to accurately extract as many fields or records as possible from a collection of unstructured or semi-structured text documents. In this scenario, we assume that we already have a partial database and we need only fill in its holes. In this chapter, we propose methods for finding specific information in a large collection of external documents, and doing so efficiently with limited computational resources. For instance, this small piece of information may be a missing record, or a missing field in a database that would be acquired by searching a very large collection of documents, such as the Web. Using traditional models of information extraction for this task is wasteful, and in most cases computationally intractable. A more feasible approach for obtaining the required information is to automatically issue appropriate queries to the external source, select a subset of the retrieved documents for processing and then extract the specified field in a focussed and efficient manner. We can further enhance the efficiency of our system by exploiting the inherent relational nature of the database. We call this process of searching and extracting for specific pieces of information, on demand, *Resource-bounded Information Extraction* (RBIE). In this chapter, we present the design of an early framework for Resource-bounded Information Extraction, discuss various important design choices involved and present some experimental results.

Consider a database of scientific publication citations, such as Rexa, Citeseer or Google Scholar. The database is created by crawling the web, downloading papers, extracting citations from the bibliographies and then processing them by tagging and normalizing. In addition, the information from the paper header is also extracted. In order to make these citations and papers useful to the users, it is important to have the year of publication information available. Even after integrating the citation information with other publicly available databases, such as DBLP, a large fraction of the papers do not have a year of publication associated with them. This is because, often, the headers or the full text of the papers do not contain the date and venue of publication (especially for preprints available on the web). Approximately one third of the papers in Rexa are missing the year of publication field. Our goal is to fill in the missing years by extracting them from the web.

Note that, in the setting described above, we are often not interested in obtaining the complete records on the database, but in just filling in the missing values. Also, the corpus of documents, such as the web, is extremely large. Moreover, in most real scenarios, we must work under pre-specified resource constraints. Any method that aims to extract required information in the described setting must be designed to work under the given resource constraints. Hence, this is a good example of an RBIE problem.

Many of these databases are relational in nature, for example, obtaining the value of one field may provide useful information about the remaining fields. Similarly, if the records are part of a network structure with uncertain or missing values, as in the case of the citation network in our example task, then information obtained for one node can reduce uncertainty in the entire network. We show that exploiting these kinds of dependencies can reduce the amount of resources required to complete the task significantly. The *db-inference* action in this case involves propagating information obtained from the external source through the graph, so as to reduce uncertainty about the values of each entry.

5.2 General Problem Setup

The previous two chapters primarily focused on selecting *query* actions of the RBIA framework in Chapter 1, and how it is influenced by *db-inference.* We now expand our focus on other aspects of the framework, by also incorporating *download* and *extract* actions. Note that RBIE is again, an instantiation of the general RBIA framework, in that the external information to be acquired is embedded in semi-structured or unstructured documents, and must be extracted before use. We now present a general problem setup for which the methods proposed in this chapter may be applicable.

Let DB be a database with a set of instances I. Let X_e be the set of fields with existing values, and x_m be a field with missing values, which we want to acquire. We assume that the values in X_e can be used as an input for issuing queries to an external information source, such as the Web, that potentially contains the missing values for x_m. We assume that we have a probabilistic model for extracting values of x_m from semi-structured or unstructured documents obtained from the external source. Finally, we assume that there exists a temporal partial order over all the instances in I, imposed by the values of x_m. This last assumption is exploited by the information propagation methods in this chapter for reducing the amount of resources required for information acquisition. Note that the methods for information propagation proposed in this chapter are specifically designed for the case of temporal partial order, and may not work for a general graph structure. Extending these ideas for a general directed or undirected graph structure is part of our future work.

5.3 System Architecture

We need a new framework for performing information extraction to automatically acquire specific pieces of information from a very large corpus of unstructured documents. Fig. 5.1 shows a top-level architecture of our proposed framework. This is

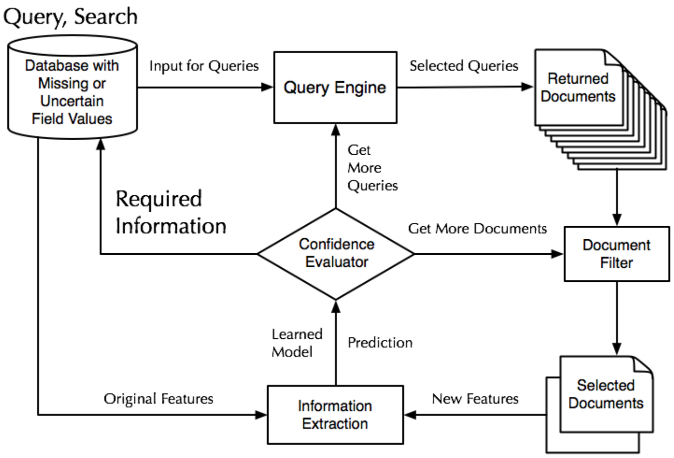


Figure 5.1. General Framework for Resource-bounded Information Extraction

an early framework used for an RBIE task, and even though it is fairly general in terms of the components of such as system, it does not provide a general method for selecting actions. In the next chapter, we will see further generalization of our framework.

In this section, we discuss the general ideas for designing a resource-bounded information extraction system. Each of these modules may be adapted to suit the needs of a specific application, as we shall see for our example task.

We start with a database containing missing values. In general, the missing information can either be a complete record, or values of a subset of the features for all records, or a subset of the records. We may also have uncertainty over the existing feature values that can be reduced by integrating external information. We assume that the external corpus provides a search interface that can be accessed automatically, such as a search engine API.

The information already available in the database is used as an input to the *Query Engine*. The basic function of the query engine is to automatically formulate queries, prioritize them optimally, and issue them to a search interface. The documents returned by the search interface are then passed on to the *Document Filter*. Document

Filter removes documents that are not relevant to the original database and ranks the remaining documents according to the usefulness of each document in extracting the required information.

A machine learning based information extraction system extracts relevant features from the documents obtained from the Document Filter, and combines them with the features obtained from the original database. Hence, information from the original database and the external source is now merged, to build a new model that predicts the values of missing fields. In general, we may have resource constraints at both training and test times. In the training phase, the learned model is passed to the *Confidence Evaluation System*, which evaluates the effectiveness of the model learned so far and recommends obtaining more documents through Document Filter, or issuing more queries through the Query Engine in order to improve the model. In the test phase, the prediction made by the learned model is tested by the Confidence Evaluation System. If the model's confidence in the predicted value crosses a threshold, then it is used to fill (or to replace a less certain value) in the original database. Otherwise, the Confidence Evaluation System requests a new document or a new query to improve the current prediction. This loop is continued until either all the required information is satisfactorily obtained, or we run out of a required resource. Additionally, feedback loops can be designed to help improve performance of Query Engine and Document Filter.

This gives a general overview of the proposed architecture. We now turn to a more detailed description for each module, along with the many design choices involved while designing a system for our specific task.

We present a concrete resource-bounded information extraction task and a probabilistic approach to instantiate the framework described above: We are given a set of citations with fields, such as, paper title, author names, contact information available, but missing year of publication. The goal is to search the web and extract this information from web documents to fill in the missing year values. We evaluate the performance of our system by measuring the precision, recall and F1 values at different confidence levels. The following sections describe the architecture of our prototype system, along with possible future extensions.

5.3.1 Query Engine

The first step in the information acquisition process is requesting the external information, or the location thereof. The basic function of query engine is to automatically formulate queries, prioritize them optimally, and issue them to a search interface. There are three modules of query engine. The available resources may allow us to acquire the values for only a subset of the fields, for a subset of the records. Input selection module decides which feature values should be acquired from the external source to optimize the overall utility of the database. The query formulation module combines input values selected from the database with some domain knowledge, and automatically formulates queries. For instance, a subset of the available fields in the record, combined with a few keywords provided by the user, can form useful queries. Out of these queries, some queries are more successful than others in obtaining the required information. Query ranking module ranks the queries in an optimal order, requiring fewer queries to obtain the missing values. In the future, we would like to explore sophisticated query ranking methods, based on the feedback from other components of the system.

In our system, we use existing fields of the citation, such as paper title and names of author, and combine them with keywords such as “cv”, “publication list”, etc. to formulate the queries. We experiment with the order in which we select citations to query. In one method, the nodes with most incoming and outgoing citation links are queried first. We issue these queries to a search API and the top n hits (where n depends on the available resources) are obtained.

5.3.2 Document Filter

Even though queries are formed using the fields in the database, some documents may be irrelevant. This may be due to the ambiguities in the data (e.g. person name coreference), or simply imperfections in the retrieval engine. We need a mechanism to remove such irrelevant documents. The primary function of the document filter is to remove irrelevant documents and prioritize the remaining documents for processing. Following are the two main components of the Document Filter. Initial filter removes documents which are irrelevant to the original database. The remaining documents are then ranked by document ranker, based on their relevance to the original database. Remember that the relevance used by the search interface is with respect to the queries, which may not necessarily be the same as the relevance with respect to the original database. In the future, we would like to learn a ranking model, based on the feedback from the information extraction module (via *Confidence Evaluation System*) about how useful the document was in making the actual prediction.

In our system, many of the documents returned by the search engine are not relevant to the original citation record. For example, a query with an author name and keyword "resume" may return resumes of different people sharing a name with the paper author. Hence, even though these documents are relevant to an otherwise useful query, they are irrelevant to the original citation. Sometimes, the returned document does not contain any year information. The document filter recognizes these cases by looking for year information and soft matching the title with body of the document.

5.3.3 Probabilistic prediction model for Information Extraction

Next, we need a method for extracting the required information from web documents. However, the design of this module differs from traditional information extraction, posing interesting challenges. We need a good integration scheme to merge features from the original database with the features obtained from the external source. As new information (documents) arrives, the parameters of the model need to be updated incrementally (at train time), and the confidence in the prediction made by the system must be updated efficiently (at test time).

In our task, the field with missing values can take one of a finite number of possible values (i.e. a given range of years). Hence, we can view this extraction task as a multi-class classification problem. Features from the original citation and web documents are combined to make the prediction using a maximum entropy classifier.

Let c_i be a citation ($i = 1, \ldots, n$), q_{ij} be a query formed using input from citation c_i and d_{ijk} be a document obtained as a result of q_{ij}. Assuming that we use all the queries, we drop the index j. Let y_i be a random variable that assigns a label to the citation c_i. We also define a variable y_{ik} to assign a label to the document d_{ik}. If Y is the set of all years in the given range, then $y_i, y_{ik} \in Y$. For each c_i, we define a set of m feature functions $f_m(c_i, y_i)$. For each d_{ik}, we define a set of l feature functions $f_{lk}(c_i, d_{ik}, y_{ik})$. For our model, we assume that $f_m(c_i, y_i)$ is empty. This is because the information from the citation by itself is not useful in predicting the year of publication. In the future, we would like to design a more general model that takes these features into account. We can now construct a model given by

$$P(y_{ik}|c_i, d_{ik}) = \frac{1}{Z_d} \sum_l \exp(\lambda_l f_l(c_i, d_{ik}, y_{ik})), \tag{5.1}$$

where $Z_d = \sum_y \exp(\lambda_l f_l(c_i, d_{ik}, y_{ik}))$

The above model outputs y_{ik} instead of the required y_i. We have two options to model what we want. We can either merge all the features $f_{lk}(c_i, d_{ik}, y_{ik})$ from d_{ik}'s to form a single feature function. This is equivalent to combining all the evidence for a single citation in the feature space. Alternatively, we can combine the evidence from different d_{ik}'s in the output space. Following are two of the possible schemes for combining the evidence in the output space. In the first scheme, we take a *majority* *vote*, i.e., the class with the highest number of y_{ik} is predicted as the winning class and assigned to y_i. In the second scheme, *highest confidence* scheme, we take the most confident vote, i.e., $y_i = argmax_{y_{ik}} P(y_{ik}|c_i, d_{ik})$

5.3.4 Confidence Evaluation System

At train time, the Confidence Evaluation System can measure the 'goodness' of the model after adding each new training document by evaluating it on a validation set. At test time, confidence in the prediction improves as more information is obtained. It sets a threshold on the confidence, to either return the required information to the database, or to request more information from external source. It also makes the choice between obtaining a new document or to issue a new query at each iteration, by taking into account the cost and utility factors. Finally, it keeps track of the effectiveness of queries and documents in making a correct prediction. This information is useful for learning better ranking models for Query Engine and Document Filter.

In our system, we train our model using all available resources, and focus on evaluating test time confidence. For merging evidence in the output space, we employ two schemes. In *max votes*, we make a prediction if the percentage of documents in the winning class crosses a threshold. In *highest confidence*, we make a prediction if $P(y_{ik}|c_i, d_{ik})$ value of the document with the highest P in the winning class passes a threshold. These schemes help determine if we have completed the task satisfactorily. For combining evidence in feature space, we use the *Entropy Method*, in which we compute the value $H = -\sum_i p_i \log p_i$ of the current distribution, and compare it against the confidence threshold. This is the first part of *db-inference* action.

5.4 Uncertainty Propagation in Citation Graph

The inherent dependency within the given data set can be exploited for better resource utilization. In our case, the citation link structure can be used for inferring temporal constraints. For example, if paper A cites paper B, then assuming that papers from future can't be cited, we infer that B must have been published in the same or earlier year than A. Initially, we have no information about the publication year for a citation. As information from the web arrives, this uncertainty is reduced. If we propagate this reduction in uncertainty (or belief) for one of the nodes through the entire graph, we may need fewer documents (or fewer queries) to predict the publication year of the remaining nodes. Selecting the citations to query in an effective order may further improve efficiency.

5.4.1 Propagation Methods

The method *Best Index* passes the uncertainty message to the neighbors of c as follows:

$$\forall c_b \in C_B P_{c_b}(X = x) = P(X = x | x \geq y) \tag{5.2}$$

$$\forall c_a \in C_A P_{c_a}(X = x) = P(X = x | x < y) \tag{5.3}$$

Where $y = argmax_y P'_c(X = y)$. $P(X = x|x \geq y)$ and $P(X = x|x < y)$ are given by one of the update methods described below. The method *Weighted Average* takes a weighted average over all possible $y's$:

$$\forall c_b \in C_B P_{c_b}(X = x) = P'_c(X = y) \sum_y P(X = x | x \geq y) \tag{5.4}$$

$$\forall c_a \in C_A P_{c_a}(X = x) = P'_c(X = y) \sum_y P(X = x | x < y) \tag{5.5}$$

5.4.2 Update Methods

If we know that the given paper was published after a certain year, then we can set the probability mass from before the corresponding index to zero and redistribute it to the years after the index. We only show update in one direction here for brevity. The first update method, *Uniform Update*, simply redistributes the probability mass, $P(x \geq y)$ uniformly to the remaining years. The second update method, *Scale Update*, uses conditional probability.

$$\begin{aligned} P(X = x|x \geq y) &= 0, x < y && (5.6) \\ &= P(X = x) + \frac{1}{P(x \geq y)}, x \geq y && (5.7) \end{aligned}$$

$$\begin{aligned} P(X = x|x \geq y) &= 0, x < y && (5.8) \\ &= \frac{P(X = x)}{P(x \geq y)}, x \geq y && (5.9) \end{aligned}$$

5.4.3 Combination Methods

Along with passing a message to its neighbors, the node updates itself by combining information from the *Document Classifier* and the graph structure.

$$P_c(X = x) = P_c'(X = y) \sum_y P(X = x|x = y) \tag{5.10}$$

The following options can be used for computing $P_c(X = x)$. *Basic*, $P(X = x|x = y)$ *Product* $P_c(X = x) * P_c'(X = x)$ and *Sum* $P_c(X = x) + P_c'(X = x)$

5.5 Experimental Results

5.5.1 Dataset and Setup

Our data set consists of five citation graphs (462 citations), with years of publication ranging from 1989 to 2008. The sampling process is parameterized by size of the network (20-100 citations per graph) and density (min in-degree = 3 and min out-degree = 6). We use five-fold cross validation on these data sets for all our experiments. We use Mallet [59] for training and testing, and Google search API to issue queries. The queries formed using the information from input citations include the raw title, title in quotes, and author names combined with keywords like “publication list”, “resume”, “cv” , “year” and “year of publication”. We issue queries in a random order, and obtain top 10 hits from google. We use around 7K queries and obtain around 15K documents after filtering. The documents are tokenized and tokens are tagged to be possible years using a regular expression. The document filter discards a document if there is no year information found on the webpage. It also uses a soft match between the title and all n-grams in the body of the page, where n equals the title length. If there is at least one n-gram with more than 75% overlap with title tokens, then the document is retained. The selected documents are passed on in a random order to the MaxEnt model, which uses the following features for classification: Occurrence of a year on the webpage; the number of unique years on the webpage; years on the webpage found in any particular order; the years that immediately follow or precede the title matches; the distance between a ‘surrounding’ year and its corresponding title match and occurrence of the same ‘following’ and ‘preceding’ year for a title match.

5.5.2 Results and Discussion

We first run our RBIE system without exploiting the citation network information. Table 5.1 shows the results for combining evidence in the feature space. We measure Precision, Recall and F1 based on using a confidence threshold, where F1 is the harmonic mean of precision and recall. As seen in table 5.1, as we increase the entropy threshold, precision drops, as expected. F1 peaks at threshold 0.7. Note that the number of documents is proportional to the number of queries, because in our experiments, we stop obtaining more documents or issuing queries when the threshold is reached.

Entropy Threshold	Precision	Recall	F1	#Queries	#Docs	Fraction of Docs
0.1	0.9357	0.7358	0.8204	4497	9564	63.76%
0.3	0.9183	0.8220	0.8666	3752	8010	53.40%
0.5	0.9013	0.8718	0.8854	3309	7158	47.72%
0.7	0.8809	0.9041	0.8909	2987	6535	43.56%
0.9	0.8625	0.9171	0.8871	2768	6088	40.58%

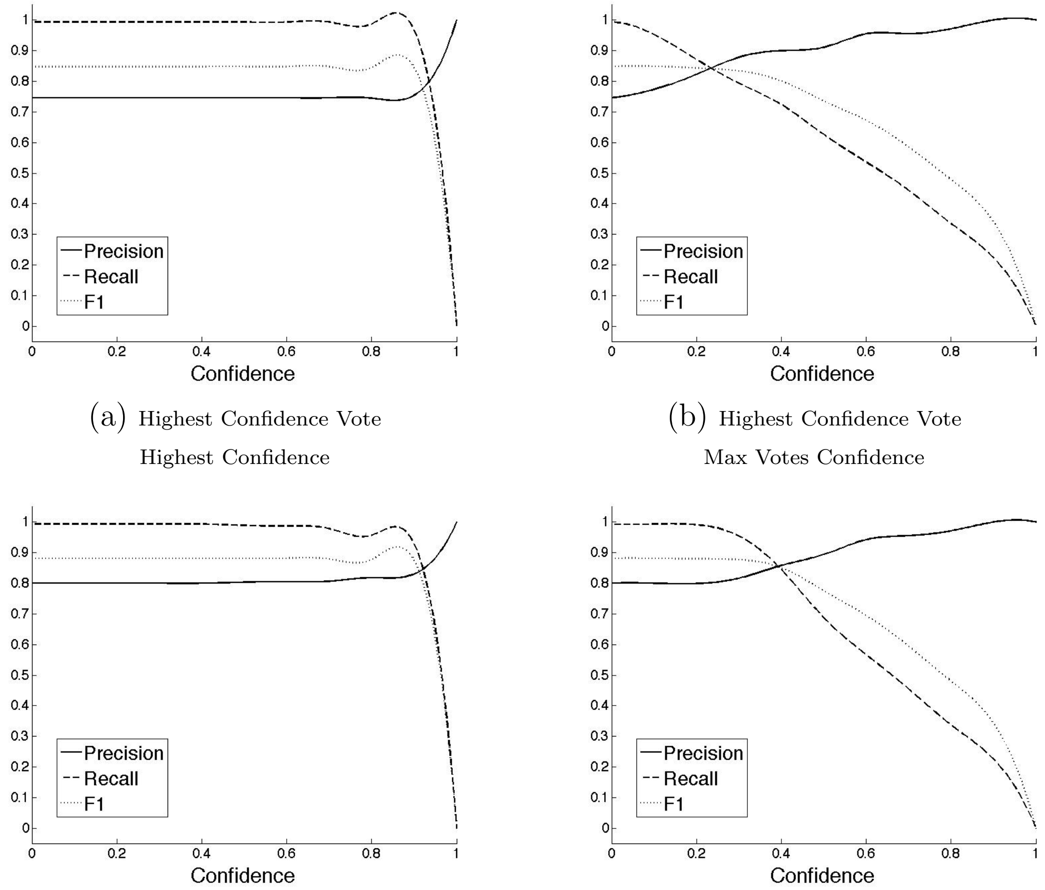
Table 5.1. Baseline results. The graph based method (*Weighted Avg* propagation, *Scaling* update, and *Basic* combination) gives an F1 value of 0.72 using only 3.06% documents at all threshold levels.

Update	Combination	F1 for Best Index	F1 for Weighted Avg
Uniform	Basic	0.7192	0.7249
Uniform	Sum	0.7273	0.5827
Uniform	Product	0.6475	0.3460
Scaling	Basic	0.7249	0.7249
Scaling	Sum	0.6875	0.5365
Scaling	Product	0.6295	0.4306

Table 5.2. Comparison of Uncertainty Propagation Methods

Next, we present the results for exploiting citation network information for better resource utilization. Table 5.2 shows the F1 values obtained using different uncertainty propagation methods at entropy threshold 0.7. The F1 values are smaller compared to the baseline, because we use far fewer resources, and the uncertainty propagation methods are not perfect. Using this method, we are able to achieve 87.7% of the baseline F1, by using only 13.2% of the documents compared to the corresponding baseline result (at threshold 0.7). In absolute terms, the graph based method (*Weighted Avg* propagation, *Scaling* update, and *Basic* combination) gives an F1 value of 0.72 using only 3.06% of the total documents. This demonstrates the effectiveness of the information propagation methods and the value of exploiting relational nature of the data for RBIE. In the future, belief propagation like methods can be applied to this problem.

We also experiment with combining evidence in the output space using the two schemes, and the confidence evaluation schemes described in section 5.3. Fig. 5.2



(a) Highest Confidence Vote
Highest Confidence

(b) Highest Confidence Vote
Max Votes Confidence

(c) Majority Vote
Highest Confidence Vote

(d) Majority Vote
Max Votes Confidence

Figure 5.2. Different combinations of voting and confidence evaluation schemes.

shows the four precision-recall curves. We see that for *High Confidence Confidence* evaluation scheme (fig. 5.2(a),(c)), we obtain high values of precision and recall for reasonable values of confidence. That is, in the confidence region below 0.9, we obtain a good F1 value. Especially, the *Majority Vote - High Confidence* scheme (fig. 5.2(c)) performs exceptionally well in making predictions. However, in the confidence region between 0.9 to 1.0, the *Max Vote* scheme (fig. 5.2(b),(d)) gives a better degradation performance.

5.6 Chapter Summary

This chapter gives the formal definition of the problem of Resource-bounded Information Extraction (RBIE), and proposes a new framework for targeted information extraction under resource constraints to fill missing values in a database. We present first results on an example task of extracting missing year of publication of scientific papers, and show how information acquired from an external source can be propagated through a graph, so that uncertainty about the neighbors of an input instance is reduced, requiring fewer resources. The specific methods recommended here can also be generalized in many different relational domains, especially when the dataset has an underlying network structure. In future, we would like to explore more sophisticated uncertainty propagation methods, such as belief-propagation. We can also explore methods for effectively selecting (say, highly connected) nodes in the graph for querying. Finally, it would be interesting to see how these methods extend to extracting multiple interdependent fields.

Under this framework, one way to improve action selection methods is by developing individual components like *Query Engine* and *Document Filter*, by using good ranking procedures. However, as we have seen, information acquisition methods interact significantly with each other, and hence, we need an RBIE framework that selects the best action from all types of available actions at each point of time. Also, majority of the resource savings in this example come from exploiting the relational nature of the data, and we would like to instead have a more general purpose resource saving framework, that also works for i.i.d. instances. We will see such a framework in the next chapter.

CHAPTER 6

LEARNING TO SELECT ACTIONS FOR RESOURCE-BOUNDED INFORMATION EXTRACTION USING REINFORCEMENT LEARNING

6.1 Introduction

In the previous chapter, we looked at a basic framework for Resource-bounded Information Extraction (RBIE). The primary technique employed for saving computational resources was exploiting the interdependency of input data. However, in many RBIE applications, the input data may not exhibit such relational properties, and may instead be i.i.d. We need a general framework that is applicable even in such scenarios. One possible direction for saving resources in the RBIE framework proposed in the previous chapter is to introduce sophisticated methods to rank the list of queries and documents. The problem with this approach is that as we have seen so far, in most scenarios, the different information acquisition methods interact with each other significantly. For example, after inspecting a few non-promising documents downloaded and processed as a result of one query, the system may decide to issue a new query. Independent ranking mechanisms in individual components may not be able to sufficiently capture these interactions. What we need instead, is a general purpose, dynamically adapting, holistic framework, that takes into account the state of the database, the results of all the actions so far, as well as the properties of each action before selecting the 'best' action at each point of time. In this chapter, we propose such a general purpose framework for RBIE.

Consider the following example of a real world RBIE task. Given a database of top Computer Science departments in the United States (Table 6.1). Such a database

Univ. Name	Fall Deadline	Homepage	Faculty Dir	Num Faculty	Num Grad.
Stanford	Dec. 13, 2011	?	?	?	550
MIT	Dec. 15, 2011	?	?	?	890
Princeton	Dec.15, 2010	?	?	?	100
UC-Berkeley	Dec. 16, 2010	?	?	?	222
CMU	Dec. 15	?	?	?	?

Table 6.1. Example Database of Top Computer Science Departments in the U.S.

may compile a lot of relevant information about the departments, such as location, admission and course information, statistics about the faculty and student body, etc. The faculty directory on the department websites are often a useful resource to obtain more information about the faculty, and it is desirable to be able to point the users of such a database directly to this page. It would also be a very useful starting point for automatic extraction of more detailed information about the faculty (such as the number of faculty, research interests, etc). One way to obtain this information is to find the home pages of the departments and crawl the entire site to find the faculty directory pages. However, most department websites are large and complex, requiring us to process thousands of documents, making it a resource-intensive task.

Consider another related example. We are building a database of all faculty across departments at a university as shown in Table 6.2. We have names of the faculty, but some of the other information such as contact details, job titles and department affiliations are missing. Surprisingly, in some cases, the university administration does not have such a comprehensive, university-wide database. This may be due to the lack of data exchange, joint appointments across departments, changing contact details, etc. Building such a database would be extremely useful, since it maintains up-to-date records of the faculty, and fosters collaboration across departments. A large portion of this information exists on the Web, but it may not always be found on faculty home pages. Lecturers and faculty in some of the departments do not have home pages, and their information is sometimes scattered around the Web. Finding this information

Faculty Name	Phone	Email	Job Title	Department Name
Andrew McCallum	(413) 545-1323	?	Professor	Computer Science
Jerrold S. Levinsky	?	?	Lecturer	Legal Studies
Edward G. Voigtman	?	?	?	?
Robert W. Paynter	?	?	?	Anthropology

Table 6.2. Example Database of University Faculty

can be challenging, since it is not available in a uniform, structured manner. There are other problems such as name ambiguities and incorrect or incomplete data.

Again, we can obtain this information by crawling all the websites under the university domain. However, this, by itself is a resource-intensive task, since most university websites are large and complex, and we would need to use a lot of computational power to crawl and download the pages, along with the corresponding network bandwidth, and disk space for storing them. We would also lose out on all the information that is scattered on the Web, outside the university domain. Can we accomplish these tasks using a much smaller fraction of these resources?

We know that the information missing in the database is available on some relatively small number of pages on the Web. We need to run some extraction algorithms on those pages in order to obtain the required information. But before we can run extraction, we need to download them to our computing infrastructure, and before we can download them, we need to know where they are located on the Web. A search engine API, such as Google can help us retrieve these web pages. We first formulate queries driven by information that is already available in the database, issue them to the search interface, obtain the location of the web pages, and download them. Then we can run the necessary algorithms to extract information, and use it to fill missing entries in the database. This process is more efficient than indiscriminate processing, and would use relatively smaller amounts of resources.

RBIE for the Web as described in the previous chapter, works as follows. Queries are formed by combining existing, relevant information in the database with user defined keywords. All such queries are issued to the search API, and all of the result documents are downloaded. The resource savings come from selecting a subset of the web documents to process by exploiting the network structure in the data. In general, we may need multiple queries to obtain information about a single entry in the database, and some queries work better than others. In our university faculty example, we may form different queries with keywords such as “curriculum vitae” or “home page”, and it may be the case that one of them is often more successful than the other in finding the information we need. In some cases, the information in different fields may be interdependent, and finding one before another may be more efficient. In order to make the best use of available resources, we need to issue the most effective queries first.

In most scenarios, one only need process a subset of the documents returned by the queries. We need to know which of the search results are most likely to contain the information we are looking for. Information returned in the search result snippet can be exploited to decide if a web page is worth downloading. Similarly, some preliminary observation of the downloaded document can be useful to decide if it is worth passing through an expensive extraction pipeline. Hence, instead of viewing RBIE as selecting a subset of documents to process, we view it as a sequential decision making task with a series of resource-consuming actions, and a mechanism to select the best action to perform at each time step.

In this chapter, we formulate the RBIE problem formally as a Markov Decision Process (MDP), and propose the use of reinforcement learning techniques for solving it. The *state* of this MDP is the state of the database at each time step, and *action* is any act that leads to obtaining the required information, such that performing the action in one state leads to a different state. RBIE process is then finding the optimal policy in this MDP, so as to obtain most information with the given budget of actions, since we assume that actions consume resources. In RBIE from the Web context, actions are *query*, which is issuing a query to a search API, *download*, which is downloading a web document, and *extract*, which runs an actual extraction algorithm on a document. We assume uniform cost for each type of action in this work, but the proposed framework can easily be extended by incorporating a specific cost model for the actions and assigning the budget accordingly. By formulating RBIE for the Web as an MDP, we can explore the rich methods of optimal action selection offered by reinforcement learning.

In the RBIE for the web setting, query actions might not lead to immediate reward, but they are necessary to perform before download and extract actions, which may lead to positive rewards. Hence, we need a method that models delayed rewards effectively. We propose the use of temporal difference q-learning to learn the value function for selecting the best action from a set of alternative actions, given a certain state of the database. We also explore a fast, online, error driven algorithm, called SampleRank to learn this value function. Since both SampleRank and q-learning are novel approaches for the RBIE framework, we compare their relative performance on two example tasks. The first one is finding the URL of faculty directories of top computer science departments, and the other is finding emails, job titles and department affiliation for faculty in our university, which we call FindGuru.

In general, we can use any model of choice for information extraction in our framework that can extract the required information from a web page, and provide a confidence score for the extracted value. This score can be used to choose the best among the potential candidate values, and to determine whether or not an existing entry in the database should be updated by the newly extracted value. We present a simple, but novel information extraction method that can easily scale to large problem domains. The basic idea is to generate a list of potential candidate values from the web page, and using a binary classifier, such as maximum entropy, to classify them as being correct values or not, by observing features of the context in which they are found. The candidate with the maximum probability of being correct is used to fill the entry in the database.

Our experiments show that for the faculty directory finding task, both SampleRank and q-learning perform better than strong baselines. For faculty contact information finding task, the q-learning strategy performs better than the baseline action selection strategies, as well as SampleRank based approach for learning a value function. Given the large number of actions to choose from at each time step, and the size of the corresponding state space, the policy learned is impressive. On this task, the q-learning based approach is able to obtain 88.8% of the final F1, by only using 8.6% of all possible actions, demonstrating the effectiveness of our method.

6.2 General Problem Setup

Given a database, DB, with arbitrary set of entries with missing values, E. We assume that we have access to the search API of an external source of semi-structured or unstructured documents that potentially contain the missing values, such as the Web. We assume that there exists some information relevant to the entries in E, that can be used to formulate search queries to the external source. We also assume that we have established a method for extracting the specific pieces of missing information from semi-structured or unstructured documents acquired from the external source (In this work, we describe one such method). Finally, we assume that each individual action of querying the external source, acquiring a document or extracting information from it consumes some form of computational resources. The general problem of RBIE from the Web, is to select the best set of actions that lead to acquiring the missing values for entries in E, using least amount of resources.

Note that the methods proposed in this work would not be effective if the required information is not contained in a small subset of the documents, but instead distributed across a large set of documents. Also, these methods rely on the ability to examine the results of previous actions, and may not be applicable in situations in which this is not true. Furthermore, in our work, we assume uniform cost over different types of actions. Our approach needs to be extended to the case of highly skewed costs across different types of actions.

6.3 RBIE for the Web

The RBIE framework presented in this chapter is an instantiation of the RBIA framework presented in Chapter 1, adapted to the Web domain. The *db-inference* action here is morphed into maintaining the confidence for the best candidate in the database. For RBIE from the Web, we consider three different types of actions - *query*, *download*, and *extract*. A *query* action consists of issuing a single query to a web search API and obtaining a set of search results. In order to form the query, we need to use some existing information from an input record in the database and a set of keywords. A *download* action consists of downloading the web page corresponding to a single search result. Finally, an *extract* action consists of performing extraction on the downloaded webpage to obtain the required piece of information and using it to fill the slot in the original database. Note that each instantiated 'action' consists of the type of action as well as its corresponding argument, namely, what query to send for which instance, which URL to download, or what page to extract.

In the case of RBIE from the Web, the *query* actions can be initialized at the beginning of the task because we know which instances have missing fields, and the types of queries that can be used; but *download* actions and *extract* actions are generated dynamically and added to the list of available actions. That is, after a *query* action is performed, the *download* action corresponding to each of the search results is generated. Similarly, after a web page is downloaded, the corresponding *extract* action is generated. At each time point, only the actions that are instantiated can be considered as alternative valid actions to be performed. The RBIE task is to select the "best" action at each time point from a set of all valid actions.

We assume that we are given an existing model, M_e for extracting the required pieces of information from a single web page. We also assume that this model provides a confidence score for each value predicted. This score can be used to choose the best among the potential candidate values, and to determine whether or not an existing entry in the database should be updated by the newly extracted value.

6.3.1 Markov Decision Process Formulation

We cast the Resource-bounded Information Extraction problem as solving a Markov Decision Process (MDP), M, where the states represent the state of the database at a given time, along with any intermediate results obtained from the Web, and actions represent the *query*, *download*, and *extract* actions as described in the previous section. We represent state as a tuple $S_t\langle DB_t, I_t, I'_t\rangle$, where DB_t is the state of the database at time t, I_t is the list of intermediate URL results and I'_t is the list of intermediate page results obtained till time t. The MDP for RBIE is described as a tuple, $M\langle S_0, \gamma, T(S, a, S'), R(S)\rangle$, where S_0 is the initial state of the database, γ is the discount factor, $T(S, a, S')$ is the state transition probability, or the probability that action a in state S at time t will lead to state S' at time $t + 1$, and $R(S)$ is the reward function for being in state S.

6.3.2 The RBIE Algorithm

Let $V(a, S) \Rightarrow \Re$ be the value function that represents the expected utility of taking action a in state, S. Hence, the best action to select at each step is:

$$a_{t+1} = \arg\max_a V(a, S) \tag{6.1}$$

Algorithm 3 Resource-bounded Information Extraction for the Web using value function

```
Input:
    Database $DB$ with missing entries, $E_i$
    Learned value function $V(a, S)$
    Learned extraction model, $M_e$
    Time budget, $b$
Initialize all queries using keywords
$t = 0$
while $t <= b$ do
  $a_{t+1} = \arg\max_a V(a, S)$
  if $a_{t+1}$ is a query action then
    Issue query to a web search API
    Enqueue corresponding download actions
  else if $a_{t+1}$ is a download action then
    Download the web page
    Enqueue corresponding extract action
  else if $a_{t+1}$ is an extract action then
    Extract all candidate values from the web page
    Score each candidate using the model, $M_e$
    Fill the value of the best candidate in $E_i$
  end if
  $t = t + 1$
end while
```

Given a value function appropriate for the domain, Algorithm 3 summarizes the RBIE for Web framework for filling missing information in a database.

6.4 Learning the Value Function

In most real-world applications, the value function does not take a standard form, and is not readily available. Hence, it must be learned using existing data. We can represent the value function as some function of the feature values, and learn the weights on those features through parameter-learning methods. We apply two different methods to learn a value function appropriate for an RBIE task. The first one is an online, error driven algorithm, called SampleRank [18, 92], which is adapted to our state-action framework and promises to be a fast method for learning the parameters; and the other is temporal difference q-learning[90], which is one of the standard methods for learning a value function from data. In this section, we will see how these methods are applied for RBIE.

6.4.1 SampleRank for RBIE

SampleRank was first introduced in the context of learning parameters for a graphical model [18, 92]. The online nature of SampleRank lets us update the parameters for each new sampled state during the training process without the need to perform inference between each step. SampleRank also allows us to define a custom objective function, $R(S)$, which enforces *ranking constraints* between pairs of samples. We adapt it for our state-action framework to learn parameters of the value function for RBIE. In order to learn this function from training data, we first assume that its functional form is as follows:

$$V(a, S) = \exp(\sum_i \theta_i \phi_i(a, S)) \tag{6.2}$$

Where, $\Theta = \{\theta_i\}$ are model parameters and $\Phi = \{\phi_i\}$ are feature functions, defined over the the database context, the current action, and the results of all previous actions. Table 6.3 shows the notation for a quick reference.

We start training with state S_0, that represents the original state of the database. We consider all available actions at this point, and sample from states that result from these actions. In the most general version of this algorithm, we can use multiple samples at each time step to update the parameters. In our version, we only choose two samples : the state S^*, which is the result of the best action $a*$, predicted by V, and the state S', which is the best state predicted by $R(S)$.

SampleRank is an error driven learning algorithm, which lets us update parameters when the function learned up to this point makes a mistake. We say the ranking is *in error* if the function learned so far assigns a higher score to the sample with the

$V(a, S)$	Value Function for Action, a and State, S
Θ, θ_i	Parameters for V
Φ, ϕ_i	Feature Functions
$R(S)$	Objective Function
α	Learning Rate
γ	Discount Factor (for q-learning)

Table 6.3. Notation reference for learning value function from data for RBIE

lower objective, i.e.:

$$[(V_\Theta(S^*) > V_\Theta(S')) \wedge (R(S^*) < R(S'))] \vee [(V_\Theta(S^*) < V_\Theta(S')) \wedge (R(S^*) > R(S'))]$$

When this condition is true, we update the parameters, Θ using perceptron update.

$$\Theta^t \Leftarrow \Theta^{t-1} + \alpha(\phi(S'_t, a'_t) - \phi(S^*_t, a^*_t)) \tag{6.3}$$

Where, α is the learning rate used to temper the parameter updates. Note that there are also other options available for the functional form of parameter update, which are not explored here. We now choose the next best action according to value function with the new parameters and perform that action to get to the next state. Note that we can use different exploration techniques in the state space to choose the next state. We continue this process for the specified number of training iterations to obtain the final parameters of the learned value function. Algorithm 4 describes how we learn the parameters θ_i, given training data.

Under the RBIE from the Web setting, we can compute a custom objective function, $R(S)$ for state $S_t =< DB_t, I_t, I'_t >$ as a weighted sum of correct, incorrect and total number of filled values and intermediate results. The exact form of the objective function can be application specific.

Algorithm 4 SampleRank Estimation

```
Input: Training database $DB$
Initial parameters $\Theta$
Value Function $V_{\Theta}(a, S)$
Objective Function $R(S)$
$S_0 \leftarrow$ Initial State of $DB$
for $t \leftarrow 1$ to number of iterations $T$ do
    $a_t^* = \arg\max_a V_{\Theta^{t-1}}(S_{t-1}, a)$
    $S_t^* = a_t^*(S_{t-1})$
    select sample from all states $S$ reachable from $S_{t-1}$:
    $S_t' = \arg\max_S(R(S))$
    $a_t'$ = Action that led to $S_t'$
    if Ranks of $S_t'$ and $S_t^*$ assigned by $V_{\Theta^{t-1}}$ and $R$ are inconsistent then
        Update $\Theta^t \Leftarrow \Theta^{t-1} + \alpha(\Phi(S_t', a_t') - \Phi(S_t^*, a_t^*))$
    end if
    $a_t = \arg\max_a V_{\Theta^t}(S_{t-1}, a)$ //perform best action
    $S_t = a_t(S_{t-1})$
end for
```

6.4.2 Q-Learning for RBIE

One of the standard ways of solving an MDP is q-learning[90], which provides a way to learn to select the best action at each time step by using the Q-function, $Q(a, S)$. We now discuss a method to learn a q-function from real-world data. Much of the material in this section follows from [76, 83].

We know that Q-function obeys the following constraints:

$$Q(a, S) = R(S) + \gamma \sum_{S'} T(S, a, S') \max_{a'} Q(a', S') \tag{6.4}$$

To use this update equation, we need to learn the transition probability model, $T(S, a, S')$, which is difficult in our setup. Hence, we use the temporal-difference, or TD q-learning approach, which is also called model-free, because it lets us learn the Q-function without using the transition probability model. The update equation for TD q-learning is [1]:

[1] There is some disagreement amongst q-learning experts about which form of reward function to use. We choose to use $R(S')$.

$$Q(a, S) \leftarrow Q(a, S) + \alpha(R(S') + \gamma \max_{a'} Q(a', S') - Q(a, S)) \tag{6.5}$$

Where, α is the learning rate. For any real-world RBIE task, the state space for the corresponding MDP is large enough to make it very difficult to learn this function accurately. Hence, we use function approximation. We represent the Q-function as a weighted combination of a set of features as follows:

$$Q_\theta(a, S) = \sum_i \theta_i \phi_i(a, S) \tag{6.6}$$

Where $\phi_i(a, S)$ are the features of the state S and action a, and θ_i are the weights on those features that we wish to learn. We now use the following equation [76, 83] for updating the values of θ_i to try to reduce the temporal difference between successive states.

$$\theta_i \leftarrow \theta_i + \alpha \left[R(S') + \gamma \max_{a'} \hat{Q}_\theta(a', S') - \hat{Q}_\theta(a, S) \right] \frac{\partial \hat{Q}_\theta(a, S)}{\partial \theta_i} \tag{6.7}$$

We can now use this update equation to learn the parameters of our Q-function from training data. The TD-q-learning algorithm for RBIE is described in Algorithm 5. Note that we use ϵ-greedy approach for exploring the state space, where ϵ decreases in proportion to the number of training iterations.

We also need to design a custom reward function, $R(S)$ for using this algorithm. Under the RBIE from the Web setting, we can compute value of the reward function as a weighted sum of correct, incorrect and total number of filled values, number of correct candidates found, and some properties of the intermediate results and can be application specific.

6.5 The Incremental Extraction Model

In general, we can use any model of choice for information extraction in our framework that can extract the required information from a web page, and provide

Algorithm 5 Temporal difference q-learning for RBIE, with ϵ-greedy exploration

```
**Input:**
    Training database, $DB$
    Initial parameters, $\theta$
    Q Function, $Q_\theta(a, S) = \sum_i \theta_i \phi_i(a, S)$
    Reward Function, $R(S)$
    Learning Rate, $\alpha$
    Discount factor, $\gamma$
$S_0 \leftarrow$ Initial State of $DB$
**for** $t \leftarrow 0$ to number of iterations $T$ **do**
  $\epsilon = 1 - \frac{1}{T}$
  With probability $\epsilon$, pick a random action, $a_t$
  With probability $1 - \epsilon$, pick $a_t = \arg\max_a Q_{\theta_t}(a, S_t)$
  $S_{t+1} = a_t(S_t)$ //perform $a_t$
  Let $a'$ be all the valid actions from state, $S_{t+1}$
  **for** $i = 0$ to number of features **do**
    $\theta^i_{t+1} = \theta^i_t + \alpha[R(S_{t+1}) + \gamma \max_{a'} Q_{\theta_t}(a', S_{t+1}) - Q_{\theta_t}(a_t, S_t)]\phi_i(a_t, S_t)$
  **end for**
**end for**
```

a confidence score for the extracted value. In this section, we present a simple, but novel information extraction method that can easily scale to large problem domains. The basic idea is to generate a list of potential candidate values from the web page, and using a binary classifier, such as maximum entropy, to classify them as being correct values or not, by observing features of the context in which they are found. Algorithm 6 describes how we train the model.

Let E be the set of entries with missing values in the database. We use patterns and lexicons to generate a list of candidates, C_{E_i} for each entry, $E_i \in E$. A *candidate* is a unique string that is a potentially correct value for an entry in the database. Each candidate, $c_j \in C_{E_i}$, consists of a list of *mentions*, M_j, which represent the actual occurrence of the candidate string in the web documents. Each candidate may have multiple mentions, across different web pages. Corresponding to each mention, $m_k \in M_j$, we have a list of properties, or features, $f(m_k)$ which describe the context in which it was found. Since we are interested in classifying the single, canonical value of these mentions, i.e, the candidate, we collapse the properties of different mentions for a candidate c_j into a single feature function, $f(c_j)$.

Let y_{ij} be a binary variable that represents whether c_j is the correct value for entry E_i. We can then represent the probability of c_j being the correct candidate as:

$$P(y_{ij}|c_j) = \frac{1}{Z} \exp(\sum_l \lambda_l f_l(c_j, y_{ij})) \tag{6.8}$$

Where, λ_l are the weights on the features, and Z, the normalization factor is given by:

$$Z = \sum_y \exp(\sum_l \lambda_l f_l(c_j, y_{ij})) \tag{6.9}$$

Since this is a supervised approach, our training data consists of the true values of E, which can be used to train the classifier. At test time, during an extract action, we classify each $c_j \in C_{E_i}$ at that time point, and select the one with the maximum posterior probability, $P(y_{ij}|c_j)$, as the "best" candidate to fill the slot.

This incremental information extraction approach allows us to keep updating information in the database, as new Web documents are processed, making it suitable for RBIE.

6.6 Application: Faculty Directory Finding

6.6.1 Problem and Dataset Description

We are given a list of top 125 Computer Science departments in the United States, as per the 2006 ranking[1] by Computer Research Association. Our goal is to find URL of the faculty directory home page for each of these departments. This is a non-trivial task to perform in an automated fashion. Faculty directory pages of different departments have drastically different formats. They may or may not contain images

[1] http://www.cs.iit.edu/ĩraicu/rankings/CRA-CS-Rankings-1993-2006.htm

Algorithm 6 Building Extraction Model, M_e

```
Input:
   Training Database $DB$ with entries, $E$
   Pattern or Lexicon Matcher, $L(w)$ that returns a set
   of matches, $M_w$ from a Web Page, $w$
   Feature functions, $f(.)$ describing context of $M_w$
   A Supervised Learning algorithm, such as Max Ent
Initialize all queries using keywords
Initialize set of potential candidates per entry, $C_{E_i} = \{\}$
Initialize set of candidates for training, $C_t = \{\}$
while Any more actions remain do
  Pick a random action, $a$ to perform
  if $a$ is a query action, or a download action then
    Perform $a$ and enqueue corresponding download or extract actions
  else if $a$ is an extract action for Web Page, $w$ then
    $M_w \leftarrow L(w)$
    for Each match, $m_k \in M_w$ do
      if String value $(m_k)$ matches $c_j \in C_{E_i}$ then
        Add $m_k$ as a mention of $c_j$
        Merge the features, $f(m_k)$ with $f(c_j)$
      else
        Create a new candidate, $c_j$, and add to $C_{E_i}$
        $label(c_j) \leftarrow$ string value $(c_j)$ = true value $(E_i)$?
        Add $m_k$ as a mention of $c_j$
        Set $f(c_j) \leftarrow f(m_k)$
      end if
    end for
  end if
end while
for all $C_{E_i}$ do
  $C_t \leftarrow C_t \cup C_{E_i}$
end for
$M_e \leftarrow$ Train a Max Ent classifier with $f(c_t)$, for $c_t \in C_t$
```

and contact informations of faculties. It is also easy for an automated system to confuse the faculty directory page with other related pages like the faculty hiring (both types of pages almost always contain the word "faculty" in the URL) or even the home page of a particular faculty. A faculty home page may contain many names of co-authors of papers listed, contact information, as well as the word "faculty" somewhere in the URL, all of which could contribute to the mix up.

Furthermore, results of web queries are very noisy. Some of them may contain faculty directory of another university with similar name. They also tend to return the home pages of popular faculties in the department, along with some commercial websites that rank universities, and so on. Hence, we need a sophisticated model to identify faculty directory pages among all the web pages that the search interface returns.

As a precursor to our task of finding faculty directories, we find the URLs of the department home pages. This is a fairly easy task. We combine the name of the university, with keywords "computer science" to form a query and examine the top hit. In almost all cases, this returns the correct URL of the department home page. We fill the department homepage column with the returned URL.

Query Type	Query String
Q01	"*University Name* + cs + faculty directory"
Q02	"*University Name* + cs + inurl:faculty"
Q03	"*University Name* + cs + faculty site:*departmentSite*"
Q04	"*University Name* + cs + site:*departmentSite* + inurl:faculty"

Table 6.4. Types of queries for the faculty directory finding task. "cs" stands for "computer science"

The given dataset is split by 70%-30% into training and testing tests. The Google Search API is used to issue queries. For the faculty directory finding task, we formulate four different types of queries per university, as shown in Table 6.4, and consider top 20 hits returned by the search API. Assuming that we are not operating under resource-constraints, i.e., by performing all possible actions available, we get the dataset as described by Table 6.5.

Dataset	No. of Universities	No. of Queries	No. of Docs	Total Actions
Training	88	352	5941	12234
Testing	37	148	2437	5022
Total	125	500	8378	17256

Table 6.5. Datasets

6.6.2 Building the Extraction Model

Since we are looking for only the URL of the faculty directory page, rather than some other information contained within the webpage, we cast the extraction problem as a classification problem. Hence, we build a Maximum Entropy based classification model, M_e to classify each web page as a faculty directory or not. Furthermore, we use the posterior of this classifier as the confidence value for each filled entry in the database. We use MALLET [60] toolkit for building this model, and Stanford NER model for the NER features[39]. Table 6.6 describes all the features used for this model.

One of the difficulties in building this model is that there are multiple correct values of faculty directory pages. This is because pages are redirected, or web sites have multiple host names. Since it is difficult to manually label all web pages in the search results ($>$ 8000 documents), we label at most one URL from the results as the true value. Availability of more labeling resources would help improve accuracy of the model. This is because some actually correct URL could get labeled as false during the training and testing phase of the classifier and might adversely affect its performance. Note that in some cases, none of the URLs returned by the search API are correct. In such cases, we manually find one correct URL for the faculty directory page from the web and use that as the true value. There were 17 departments (13.6% of the data) for which the faculty directory page URL was not returned at all.

Features related to queries
The type of query used Hit value in the search result
Features related to URL
The document is HTML like Words like “faculty”, “directory” and “people” found Words like “job”, “hire”, “recruit”, or “employ” found A tilde sign found (might indicate a user homepage) First or last name found in non-host part of URL URL host is dot com (not a university) Same as department website Same host as department website host
Features related to Web page title
Words related to bad request found Words like “faculty”, “directory” and “people” found Words like “job”, “hire”, “recruit”, or “employ” found
Features related to Web page body
Reasonable size Phrase “bad request” or “error” found Words like “faculty”, “directory”, or “people” found Words like “phone”, “email”, “office”, or “professor” found Word “publications” found Count of Named Entities found Count of email pattern matches found Count of “PhD” pattern matches found
Features related to Web page layout
Count of images found Count of tables and cells found

Table 6.6. Features of the web page classification model for the faculty directory finding task

Let us first study the performance of the classifier, in isolation of the resource-bounded information extraction task. Any inaccuracy in this model, will not only result in poor accuracy during the RBIE process, but also mis-guide it due to inaccurate confidence prediction. Table 6.7 shows the classification performance of M_e. We also show results on the training data to show the degree of fitting of the model. Note that F1 is the harmonic mean of Precision and Recall. The main reasons of relatively lower F1 values on this model are the missing true URLs as well as the potentially inaccurate labeling as described above.

Measure	Training Set	Testing Set
Accuracy	98.38	98.07
Yes Precision	82.14	77.27
Yes Recall	72.52	61.44
Yes F1	77.03	68.45
No Precision	98.93	98.65
No Recall	99.38	99.36
No F1	99.16	99.00

Table 6.7. Performance of the web page classification model for the faculty directory finding task

6.6.3 RBIE Experiments

At test time, we start with a database that contains the university names and the home page URLs of their computer science departments. All the faculty directory URL entries are initially empty. We consider this as time, $t = 0$, and assume that each action takes one time unit. The action selection scheme that we are testing selects one of the available actions, which is performed as described in Algorithm 3. The action is then marked as completed and removed from all available actions. If an extract action is selected, it may affect the database by filling a slot and altering the confidence value associated with that slot. We evaluate the results on the database at the end of a given budget, b, or if we run out of actions.

Traditionally, information extraction literature uses different criteria for evaluating the performance of systems. We use the following definitions of evaluation metrics for our task :

$$\text{Precision} = \frac{\text{No. of Correctly Filled Entries in the Database}}{\text{No. of Filled Entries in the Database}}$$

$$\text{Recall} = \frac{\text{No. of Correctly Filled Entries in the Database}}{\text{No. of Test Entries in the Database}}$$

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

6.6.3.1 Baselines

We use two baselines for our experiments : random and straw-man. At each time step, the random approach selects an action randomly from all available actions. The straw-man approach works as follows. From our initial analysis of the results of the queries, we found that queries can be sorted by their *coverage* values as $Q03$, $Q02$, $Q04$, and $Q01$. *Coverage* of a query is the proportion of all faculty directory URLs that are contained in that query's results. This means that the first query in this order is most likely to return the correct URL. Note that this human background knowledge, and pre-processing analysis provides a huge advantage to the straw-man method. The action selection order is as follows :

- The query with the highest *coverage* value is issued for each test instance
- The first hit from the search result for each test instance is downloaded
- The first hit from the search result for each test instance is processed for extraction (classification)
- Subsequent hits from the search result for each test instance are downloaded and processed
- Subsequent queries are issued in the descending order of their *coverage* value, followed by their corresponding download and extract actions.

Note that this approach would quickly fill up the slots with the top hits of potentially effective queries, making it a very strong baseline to test our learning-based methods against.

6.6.3.2 Learning Value Function From Data

We now describe how parameters Λ for value function $V_{\Lambda}(S, a)$ are learned using training data. Table 6.8 describes the features used. Note that at train time, we do not impose resource constraints. That is, training is performed till more actions are available. However, we only run the value function learning for a given number of iterations, which acts as a type of resource constraint at training time. We determine the number of iterations and learning rate empirically.

Features related to counts
Counts of Filled Entried Counts of Intermediate Results Word 'Faculty' inside intermediate results
Features related to corresponding entry
Corresponding entry is empty Confidence value of the entry (binned)
Features related to query action
Type of query
Features related to download action
Type of the corresponding query Hit value in the search result URL and Title contains keywords URL and Title contains job related keywords The host is ".com" Same host as department website Same as department website
Features related to extract action
Type of the corresponding query Hit of the corresponding result URL and Title contains keywords URL and Title contains job related keywords Appropriate Size Bad request code found

Table 6.8. Features for learning value function for the faculty directory finding task

For learning the value function using SampleRank, we start with a database with the faculty directory URL column empty. We initialize the parameters to zero. At each time step, we explore all possible actions, sample the states and update the parameters as described in Algorithm 4. We then choose the next action to perform as per the updated parameters and proceed similarly for the specified number of iterations. In our early experiments, we tried the technique of parameter averaging, which is recommended in SampleRank literature, but in our case it did not prove to be very useful, since different types of actions lead to the update of different parameters. We also use the temporal difference q-learning method as described in Algorithm 5 as well as its variation - biased-q-learning.

We use the following form of the objective function to train our value function.

$$R(S_{t+1}) = C_n*n+C_k*k+C_l*l+C_d*d+C_r*r+C_{r'}*r'-C_{\bar{d}}*\bar{d}-C_{\bar{r}}*\bar{r}-C_{\bar{r'}}*\bar{r'} \quad (6.10)$$

Where, n is the number of slots filled in the database, k is the number of intermediate urls found, l is the number of intermediate web pages found, d is the number of slots filled correctly, $\bar{d}$ is the number of slots filled incorrectly, r is the number of correct URLs in the intermediate URL results, $\bar{r}$ is the number of incorrect URLs in the intermediate URL results, r' is the number of correct web pages downloaded in the intermediate page results and $\bar{r'}$ is the number of incorrect web pages downloaded in the intermediate page results.

In order to search through the space of parameters for the learning based methods, we try different learning rates, α with values 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 and 1.0. We also run the training for $T = 1000, 2000$ iterations. For SampleRank, we also use training iterations, $T = 5000, 10000$. We use discount factor, $\gamma = 0.9$. Finally, we try ten different, hand designed versions of the objective function, that emphasis a balance between precision and recall. For each learning-based method, we select the best performing version, as indicated by area under the acquisition curve.

6.6.4 Results And Discussion

We now compare the test-time performance of the two baselines and the value function learned through different algorithms on selecting actions at each time step. We evaluate performance after each 1000 actions from 0 to 6000. Since the initial performance of RBIE systems is interesting, we also zoom into the first 1000 actions, and look at the performance at each 100 action interval. The most effective action selection scheme is the one that is fastest in achieving high values of evaluation metrics.

6.6.4.1 RBIE Using a Candidate Classifier Oracle

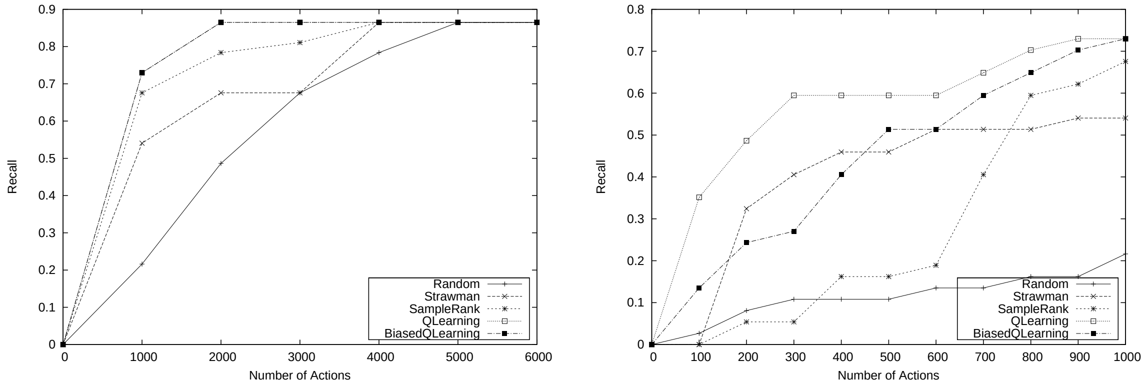


Figure 6.1. RBIE for faculty directory finding task using an oracle : Recall (figure on the right zooms to the first 1000 actions)

We first evaluate performance of the three action selection schemes in the presence of an oracle that perfectly classifies each webpage as a faculty directory page or not with infinite confidence, by looking up its true label. We do this to isolate the effect of inaccuracies in the classification model, M_e, which can severely misguide the RBIE system with wrong confidence values. For example, even if the action selection scheme selects a good web page for extraction, M_e can assign a very low confidence value to it and hence discourage updating the value in the corresponding slot. Similarly, a wrong URL with a high confidence could replace a correct one in the database slot. Table

6.7 shows that the F1 value for 'yes' label in the classifier is only 68.45, which may not be high enough to avoid some of these problems. The experiments with an oracle allow us to evaluate how well does the learned value function performs in selecting potentially useful actions early on. Note that for this experiment, we do not consider precision (and F1), since the value of precision is always one in the presence of an oracle. Hence, we know that each entry filled in the table is correct, and the scheme that obtains higher recall during the early actions has been successful in identifying the best webpages to process using fewer resources.

Figure 6.1 shows the recall values during the RBIE process for the baselines as well as proposed action selection schemes. The SampleRank method is trained with $T = 5000$ iterations, with a learning rate, $\alpha = 1.0$, and parameters for objective function, $C_n = 3000, C_k = 10000, C_l = 3000, C_d = 1000, C_r = 100, C_{r'} = 10, C_{\bar{d}} = 200, C_{\bar{r}} = 5, C_{\bar{r}'} = 1$. The q-learning and biased-q-learning are both trained with $T = 2000$ and $\alpha = 0.005$ and $\alpha = 0.05$ respectively. The parameters for q-learning are the same as those for SampleRank, and the ones for biased-q-learning are $C_n = 3000, C_k = 10, C_l = 30, C_d = 1000, C_r = 100, C_{r'} = 10, C_{\bar{d}} = 200, C_{\bar{r}} = 5, C_{\bar{r}'} = 1$. Note that the graphs for q-learning and biased q-learning in the graph on the left are overlapping. Figure on the right shows detailed differences in the first 1000 actions.

As we see, the learning based approaches perform better than the baselines. Q-learning and biased-q-learning methods are statistically significantly better than both baselines (as per Kolmogorov-Smirnov Test, p=0.2). The straw-man method is extremely effective, because it knows to process the top hits for a good query for each entry first. Given the complexity of the action domain, and the size of the state space, this policy is very difficult to learn.

To gain some intuition about the policy learned by q-learning, let us look at a few top features for each action type. These include presence of large number of intermediate results, and use of key words like 'faculty' in the URL (for download and extract actions). Interestingly, the optimum order in which the queries should be executed (as found by our 'coverage' analysis), is independently 'discovered' by the q-learning method. Hence, even though the straw-man method has the advantage of background human knowledge, such as the importance of hit value in the search engine results, the learning-based methods learn new and more elaborate patterns that show relative usefulness of different types of queries and help identify promising documents to process by 'examining' them.

6.6.4.2 RBIE Using Classification Model

We now study the performance of our proposed method using an actual classification model, M_e. In this case, each action selection strategy needs to balance both precision and recall. All learning-based methods are trained for 1000 iterations. SampleRank is trained with learning rate, $\alpha = 0.5$, and objective function parameters, $C_n = 5000, C_k = 3000, C_l = 3000, C_d = 1000, C_r = 100, C_{r'} = 10, C_{\bar{d}} = 200, C_{\bar{r}} = 0.5, C_{\bar{r'}} = 0.01$. Q-learning is trained with $\alpha = 0.01$, and parameters, $C_n = 300, C_k = 0, C_l = 0, C_d = 100, C_r = 10, C_{r'} = 10, C_{\bar{d}} = 200, C_{\bar{r}} = 0.5, C_{\bar{r'}} = 0.5$. Biased-q-learning is trained with $\alpha = 0.1$, and parameters, $C_n = 3000, C_k = 10000, C_l = 3000, C_d = 1000, C_r = 100, C_{r'} = 10, C_{\bar{d}} = 200, C_{\bar{r}} = 5, C_{\bar{r'}} = 1$.

Figure 6.2 shows that the straw-man approach is better at achieving high recall early on, but learning based methods are better at selecting more useful web pages to be able to fill the slots more accurately. This lets SampleRank to obtain most of the F1 value within the first 100 actions, and outperform the baselines in the first, crucial 400 actions. The drop in precision later on is due to inaccuracies in the confidence values predicted by M_e, which leads to a correct entry being replaced by an incorrect one.

Biased-q-learning is able to obtain the best F1 value it can achieve using only 800 actions (which is around 15% of all actions). This demonstrates the effectiveness of

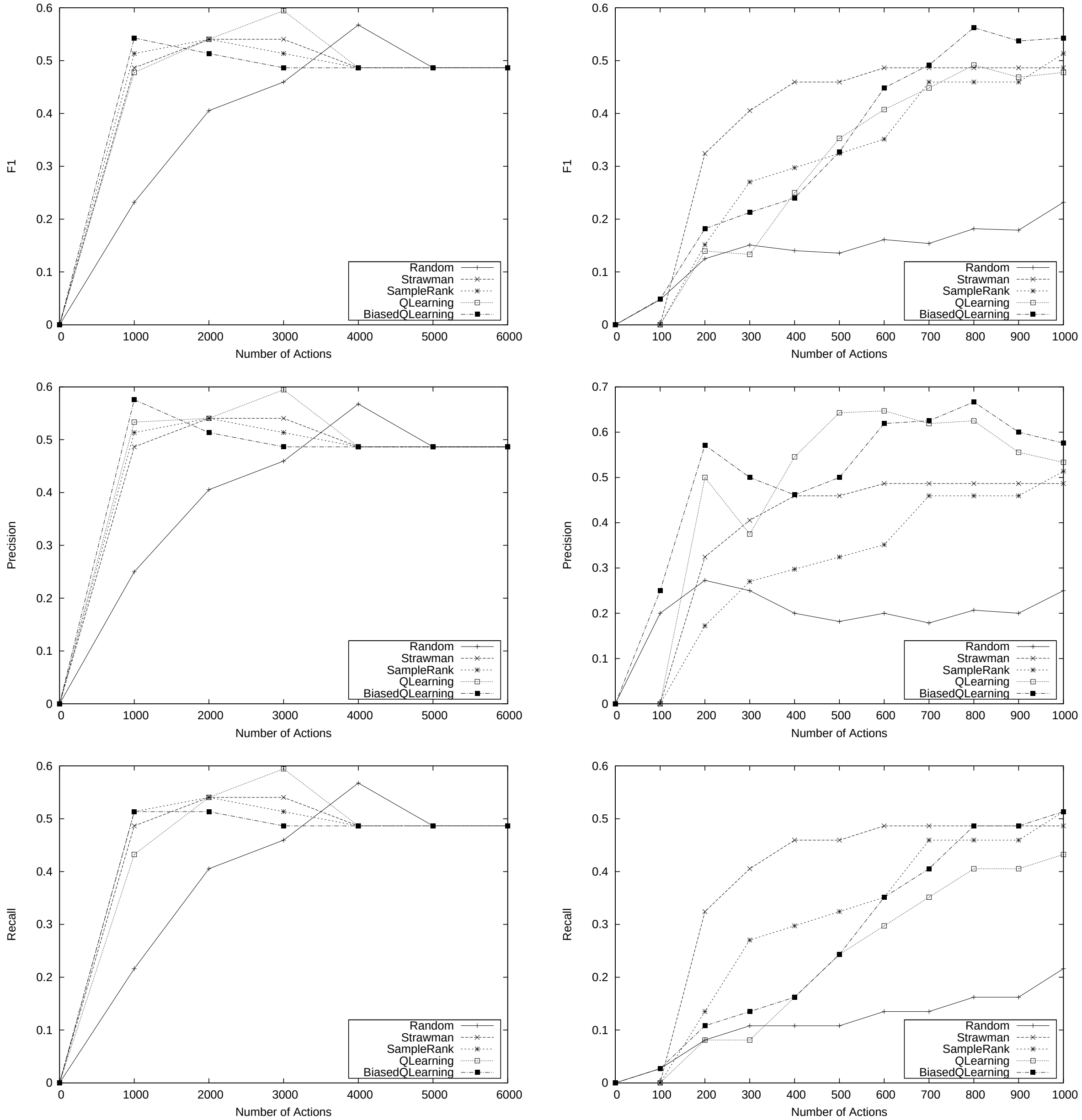


Figure 6.2. RBIE for faculty directory finding task using the classification model, M_e : From top, F1, Precision, Recall (figures on the right zoom to the first 1000 actions)

the learning-based approach in selecting good actions for information gathering task. We believe that with more accurate labeling and a better classifier, learning-based method can be shown to be even more efficient.

6.7 Application: FindGuru, Extracting Faculty Information

6.7.1 Problem Setup

Given a list of names of university faculty, our goal is to extract their email address, job title and department affiliation from the Web. In this section, we describe how we apply the RBIE framework to build a system that can efficiently acquire this information. We also describe experiments testing the effectiveness of SampleRank and q-learning algorithms in selecting the most effective actions at each time step.

This is a challenging task due to several factors. In some cases, this information is readily available on faculty home pages, which are semi-structured. However, lecturers and faculty in many departments do not have home pages. Their information is scattered around the Web, without a uniform structure. Web pages are noisy, and may lead to unexpected errors while performing extraction. Name ambiguity is another challenge, since many of the faculty have common names they share with other famous personalities. Some information on the Web is stale, or contradicting. For example, a faculty member can be listed on one page as "assistant professor", while on another as "associate professor", reflecting a recent change of title. Finally, some information is not available on the Web at all.

6.7.2 Dataset Description

We start with a list of faculty from University of Massachusetts at Amherst. We randomly choose 100 of these records as our dataset. The fields contain the first, middle and last name of the faculty, their email address, a list of job titles, and a list of department affiliations. Joint appointments lead to multiple job titles and department affiliations. The dataset we received contains several inaccuracies and is cleaned for better evaluation of our methods. For example, in some cases, a single column contains names of different departments. These are split into multiple columns. Punctuation and abbreviations, such as “Assoc. Prof.” are cleaned and expanded. Despite the cleaning effort, the dataset we use is incomplete and contains errors. For example, the most current job titles are not reflected, and only one email address is included in the dataset, which may not be the one used by the person, or published on the Web. These imperfections in the data make both training and evaluation of our system challenging. Another problem in evaluating the accuracy of our system is the “generic-specific” problem in department names. For example, our system might predict the department affiliation for a faculty as “finance”, while it might be listed as “management” in the ground truth dataset, or vice-versa. Since we use exact string match for evaluation, we may even miss a match such as “school of management”. Despite the difficulties, it is an interesting real world task for RBIE.

Name	Name + Univ
Name in quotes	Name in quotes + Univ
Name In Univ	Name in quotes In Univ
Name w/ middle In Univ	Name w/ middle + Univ
Name + CV	Name + Univ + CV
Name + “Resume”	Name + Univ + “Resume”
Name + “Profile”	Name + Univ + “Profile”
Name + “Bio”	Name + Univ + “Bio”
Name + HomePage	Name + HomePage In Univ
Name + “Contact”	Name + “Contact” In Univ

Table 6.9. Types of queries for FindGuru task. ‘Name’ : first and last name, ‘CV’ : “curriculum vitae”, ‘Univ’ : “university of massachusetts at amherst” and ‘In Univ’ : “site:umass.edu”

We use the Google search API for our experiments. In our task, the three fields that we extract are related to each other and often found in the proximity of each other on the same web pages. Hence, our query actions correspond to the entire record in the database, as opposed to a single ‘entry’, or cell. We formulate 20 different types of queries per faculty, as shown in Table 6.9, and consider top 20 hits returned by the search API. Assuming that we are not operating under resource-constraints, i.e., we perform all possible actions available, we get the dataset as described by Table 6.10. This table also helps estimate the size of the state space for our problem.

Dataset	# Faculty	# Queries	# Docs	Total Actions
Training	70	1400	13686	28772
Testing	30	600	6065	12730
Total	100	2000	19751	41502

Table 6.10. Datasets for FindGuru task

6.7.3 Training the Extraction Models

Before we move to the action selection experiments, we must build a model for extracting the relevant fields from individual web pages. Section 6.5 describes the algorithm we use for training the model. We use the MALLET [60] toolkit's implementation of the maximum entropy classifier. The available data is first split by 70%-30% for training and testing. The training phase for the extraction model is not resource-constrained, i.e., we use all possible query, download and extract actions.

The algorithm described in section 6.5 uses a pattern or lexicon matcher that returns a set of matches from a web page. These matches are added as a list of candidates to be filled in the database entry. For emails, we use a regular expression to match all the emails found in the web document. For job titles and department affiliations, we first build N-grams from body of the web page, where $N = 1, 2, 3, 4$. These N-grams are matched against lexicons to find candidate mentions in the web page. The features used to describe the context of these matches are shown in Table 6.11. The features across a mention are collapsed by using an 'OR' operator, since they are mostly binary. That is, if any feature is turned on in one of the mentions, it would be turned on for the candidate. In our early experiments, we found this method perform better than other merging operations, however, in future, we can build a more sophisticated method.

Let us first study the performance of the candidate classifier, in isolation of the resource-bounded information extraction task. Any inaccuracy in this model, will not only result in poor accuracy during the RBIE process, but also misguide it due to inaccurate confidence prediction. Table 6.12 shows the classification performance of M_e. Note that F1 is the harmonic mean of precision and recall. The main reasons for relatively lower F1 values on this model are inaccuracy of training data as described above, as well as the noisy nature of the web data. The advantage of using this model is that it is easy to build, and is scaleable for very large scale problems. In the future, we would like to experiment with a more sophisticated extraction model, in order to facilitate better accuracy of the classifier, as well as the RBIE process.

6.7.4 Experiments

At test time, we start with the database that contains names of faculty. All other columns are empty. We consider this as time, $t = 0$. We assume that each action takes one time unit. The action selection scheme that we are testing selects one of the available actions, which is performed as described in Algorithm 3. The action is then marked as completed and removed from all available actions. If an extraction action is selected, it may affect the database by filling a slot and altering the confidence value associated with that slot. We evaluate the results on the database at the end of a given budget, b, or if we run out of actions.

We are interested in finding the email address, job title and department affiliation, all of which can have multiple true values. Note that this also includes minor variations. Throughout our evaluations, we compare against the multiple values of a column, and declare a match if the predicted value exactly matches at least one of them. We use the following evaluation metrics to measure our system's performance.

Features for Email Extractor
Type of query used
Email domain is from UMass
Web page domain name from UMass
Email host and web page URL host match
Relative position of faculty name and email
Match between faculty name and email username
Similarity between faculty sname and email username
Features for Job Title Extractor
Too many matches found on page
Web page domain name from UMass
Web page URL contains faculty name
Position of match on the document
The words "Assistant" or "Associate" preceds match
Relative position of faculty name and job title
Features for Department Extractor
Too many matches found on page
Web page domain name from UMass
Web page URL contains faculty name
Position of match on the document
The word "Department" precedes match
Relative position of faculty name and department name

Table 6.11. Features of the Extraction Models for FindGuru task

Since our task is slightly different from a traditional information extraction task, we use the following definitions of evaluation metrics.

$$\text{Precision} = \frac{\text{No. of Correctly Filled Entries in the Database}}{\text{No. of Filled Entries in the Database}}$$

$$\text{Recall} = \frac{\text{No. of Correctly Filled Entries in the Database}}{\text{No. of Test Entries in the Database}}$$

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

Measure	Email	JobTitle	Department
Accuracy	97.97	92.50	96.94
Yes Precision	77.78	36.36	54.54
Yes Recall	87.50	15.38	44.44
Yes F1	82.23	21.62	48.97
No Precision	99.28	94.15	98.11
No Recall	98.57	98.06	98.73
No F1	98.93	96.06	98.42

Table 6.12. Performance of the Extraction Models for FindGuru task

$$\text{Extraction Recall} = \frac{\text{No. of Correct Candidates Extracted}}{\text{No. of Test Entries in the Database}}$$

Note that extraction recall measures the proportion of entries for which a true candidate value has been extracted from the web page. It may or may not get ranked as the "best" candidate. However, for the purpose of evaluating the order of selecting the query, download and extract actions, this is a useful metric. Even though at test time, it is independent of the performance of the underlying extraction model, M_e, it is still influenced indirectly by M_e through training.

6.7.4.1 Baselines

We use two baselines for our experiments : random, and straw-man. At each time step, the random approach selects an action randomly from all available actions. The straw-man approach is actually an extremely strong competitor and works as follows. The first query in the list is issued for each test instance. Next, the first hit from the search result for each test instance is downloaded and processed for extraction. Then, subsequent hits from the search result for each test instance are downloaded and processed. Finally, subsequent queries are issued in the descending order, followed by their corresponding download and extract actions. Note that this approach quickly fills up the slots with the top hits of the queries, making it a very difficult baseline to beat for learning-based methods.

6.7.4.2 Learning Q-function from Data

We now describe how parameters θ_i for Q-function $Q_\Theta(a, S)$ are learned using training data. Table 6.13 shows the features used. Note that at train time, we do not impose resource constraints. That is, training is performed till more actions are available. However, we only run Q-function parameter updates for a given number of iterations, which acts as a type of budget. We determine the number of iterations and learning rate empirically.

Similar to the test time, we start with a database with the email, job title and department name columns empty. The true values of these columns are used only to calculate the reward function. We initialize the parameters to zero. At each time step, we explore all possible actions, and update the parameters as described in Algorithm 5. We then choose the next action to perform as per the updated parameters and proceed similarly for the specified number of iterations.

We introduce a variation of q-learning, in which the policy is initialized using the straw-man approach, followed by the normal q-learning. In this case, a bias value proportional to the rank of actions proposed by the straw-man method is added to the q-function value for each state-action pair. We call this method 'biased-q-learning' in our experiments.

We use the following reward function for training.

$$R(S_{t+1}) = C_n * n + C_m * m + C_d * d + C_r * r - C_{\bar{d}} * \bar{d} - C_{\bar{r}} * \bar{r} \tag{6.11}$$

Where, n is the number of slots filled in the database, m is the number of correct candidates found, d is the number of slots filled correctly, $\bar{d}$ is the number of slots filled incorrectly, r is the number of web pages containing any correct slot value downloaded so far, and $\bar{r}$ is the number of web pages not containing any slot value downloaded so far.

In order to search through the space of parameters for the learning based methods, we try different learning rates, α with values 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 and 1.0. We also run the training for $T = 1000, 2000$ iterations. We use discount factor, $\gamma = 0.9$. Finally, we try ten different, hand designed versions of the objective function, that emphasis a balance between precision and recall. For each learning-based method, we select the best performing version, as indicated by area under the acquisition curve.

Features related to query, download and extract actions
Type of query
Features related to download and extract actions
Hit value in the search result URL is from UMass Webpage is HTML Title contains keywords Title contains faculty name
Features related to extract actions
Appropriate Size Bad request code found

Table 6.13. Features for learning using SampleRank and Q-function for FindGuru task

6.7.5 Results and Discussion

We now compare the test-time performance of the two baselines, the value functions learned using SampleRank, and q-learning on their ability to select good actions at each time step. Note that we have already evaluated performance of the extraction method, and we are now focussing on quality of the action selection strategy. We evaluate performance after each 2000 actions from 0 to 14000 (since the total number of actions at test time is 12730). The most effective action selection scheme is the one that is fastest in achieving high values of evaluation metrics.

6.7.5.1 RBIE Using a Candidate Classifier Oracle

We first evaluate performance of the four action selection schemes in the presence of an oracle that perfectly classifies each candidate as the correct value for an entry or not with infinite confidence by looking up the true label. We do this to isolate the effect of inaccuracies in the extraction model, M_e, which can severely misguide the RBIE system with wrong confidence values. For example, even if the action selection scheme selects a good web page for extraction, M_e can choose the wrong candidate for updating the value in the corresponding slot. While training the Q-function, this translates into incorrect reward values, which can severely impede learning. Table 6.12 shows that the F1 value for 'yes' label for each of the extractors are not high enough to avoid these problems.

All learning-based methods were trained for 2000 iterations. The SampleRank was trained for with learning rate, $\alpha = 1$, and objective function parameters, $C_n = 1, C_m = 0, C_d = 100, C_r = 10, C_{\bar{d}} = 200, C_{\bar{r}} = 0.5$. Q-learning was trained with $\alpha = 0.01$, and parameters, $C_n = 500, C_m = 0, C_d = 1000, C_r = 100, C_{\bar{d}} = 500, C_{\bar{r}} = 0.5$. Biased-q-learning is trained with $\alpha = 0.1$ and parameters, $C_n = 0, C_m = 0, C_d = 100, C_r = 0, C_{\bar{d}} = 100, C_{\bar{r}} = 0$. Figure 6.3 shows the recall values during the RBIE process for different fields, and the total number of entries. Note that in the presence of an Oracle, other evaluation metrics are not useful, since the precision is always 1, and the recall is the same as the extraction recall. Both q-learning and biased-q-learning comfortably beat the two baselines for email extraction task, and slightly outperforms the straw-man method on total entries. They also learn to beat the random action selection, as well as the value function learned by SampleRank. As expected, both q-learning and biased-q-learning perform better than SampleRank due to their modeling of delayed rewards, despite the use of exactly the same features.

To gain some intuition about the policy learned by q-learning, let us look at a few top features for each action type. For *query* actions : query type with just the name

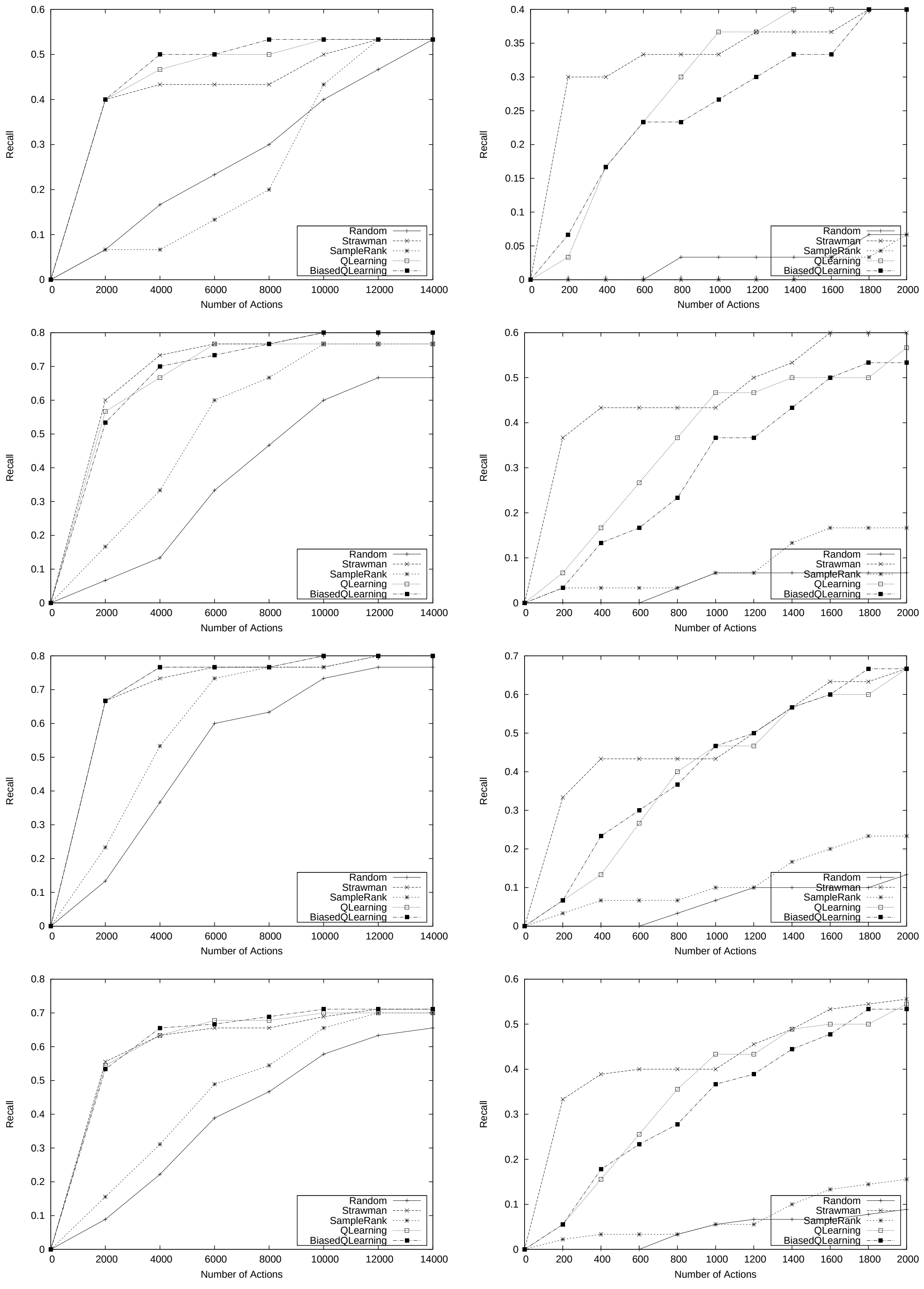


Figure 6.3. RBIE Using the Oracle for FindGuru task. The graphs from top to bottom are : Email, Job Title, Department Name and Total Entries. (figures on the right zoom to the first 2000 actions)

of the faculty, query type with the name of the faculty and the keyword, ‘Homepage’. For *download* and *extract* actions : the URL is html, URL is from a umass.edu domain, the title contains faculty name, and combinations of the corresponding query type and range of hit values. For *extract* actions, it also learned high weights for features that looked at the size of the documents. Hence, even though the straw-man method has the advantage of background human knowledge, such as the importance of hit value in the search engine results, the learning-based methods learn new and more elaborate patterns that show relative usefulness of different types of queries and help identify promising documents to process by ‘examining’ them.

Fraction of Action Budget	Fraction of Best Recall
0.00%	0.00%
1.43%	7.94%
2.86%	22.22%
4.29%	36.51%
5.71%	50.79%
7.14%	61.90%
8.57%	61.90%
10.00%	69.84%
11.43%	71.43%
12.86%	71.43%
14.29%	77.78%
28.57%	90.48%
42.86%	96.83%
57.14%	96.83%
71.43%	100.00%
85.71%	100.00%
100.00%	100.00%

Table 6.14. Effectiveness of Q-learning in obtaining recall over total entries using an oracle

Table 6.14 presents the percentage of the best extraction recall obtained over total entries for the fraction of the action budget. For example, 77.78% of the best achievable recall (using all available actions) is obtained using only 14.29% and so on. This shows that the proposed RBIE methods can be effective in obtaining most of the useful information using only a fraction of the resources. One thing to note here is that even with using oracle we are not able to achieve perfect recall at the end of al available actions. This illustrates how challenging the problem of finding information about people online is.

6.7.5.2 RBIE Using Extraction Model

We now study the performance of our proposed method using an actual extraction model, M_e. In this case, each action selection strategy needs to balance both precision and recall. We ran 1000 training iterations of SampleRank ($\alpha = 0.001, C_n = 0, C_m = 0, C_d = 100, C_r = 0, C_{\bar{d}} = 100, C_{\bar{r}} = 0$), q-learning ($\alpha = 0.01, C_n = 10, C_m = 10, C_d = 1000, C_r = 0, C_{\bar{d}} = 50, C_{\bar{r}} = 0$), and biased-q-learning ($\alpha = 0.05, C_n = 500, C_m = 0, C_d = 1000, C_r = 100, C_{\bar{d}} = 500, C_{\bar{r}} = 0.5$). Figure 6.4 shows the precision, recall, F1 and extraction recall values for total number of entries in the database, and Fig. 6.5 shows the F1 values of the individual fields. In these methods, some of the precision and recall curves go down towards the end of information gathering process due to noise in the web data, and the extraction process. As before, we see that the straw-man method performs better initially. However, its precision and recall drops mid-way through the information acquisition process, and q-learning method performs better. The F1 values over the total number of entries for both q-learning and biased-q-learning methods are statistically significantly better than both baselines (as per Kolmogorov-Smirnov Test, p=0.05). Q-learning also comfortably out-performs SampleRank approach. It achieves 88.8% of the final F1, by only using 8.6% of the total actions. This demonstrates the effectiveness of the policy learned by the Q-learner for selecting good actions for information gathering task. Note that these methods perform differently for individual fields. This demonstrates the need to tackle these fields separately during the information gathering process.

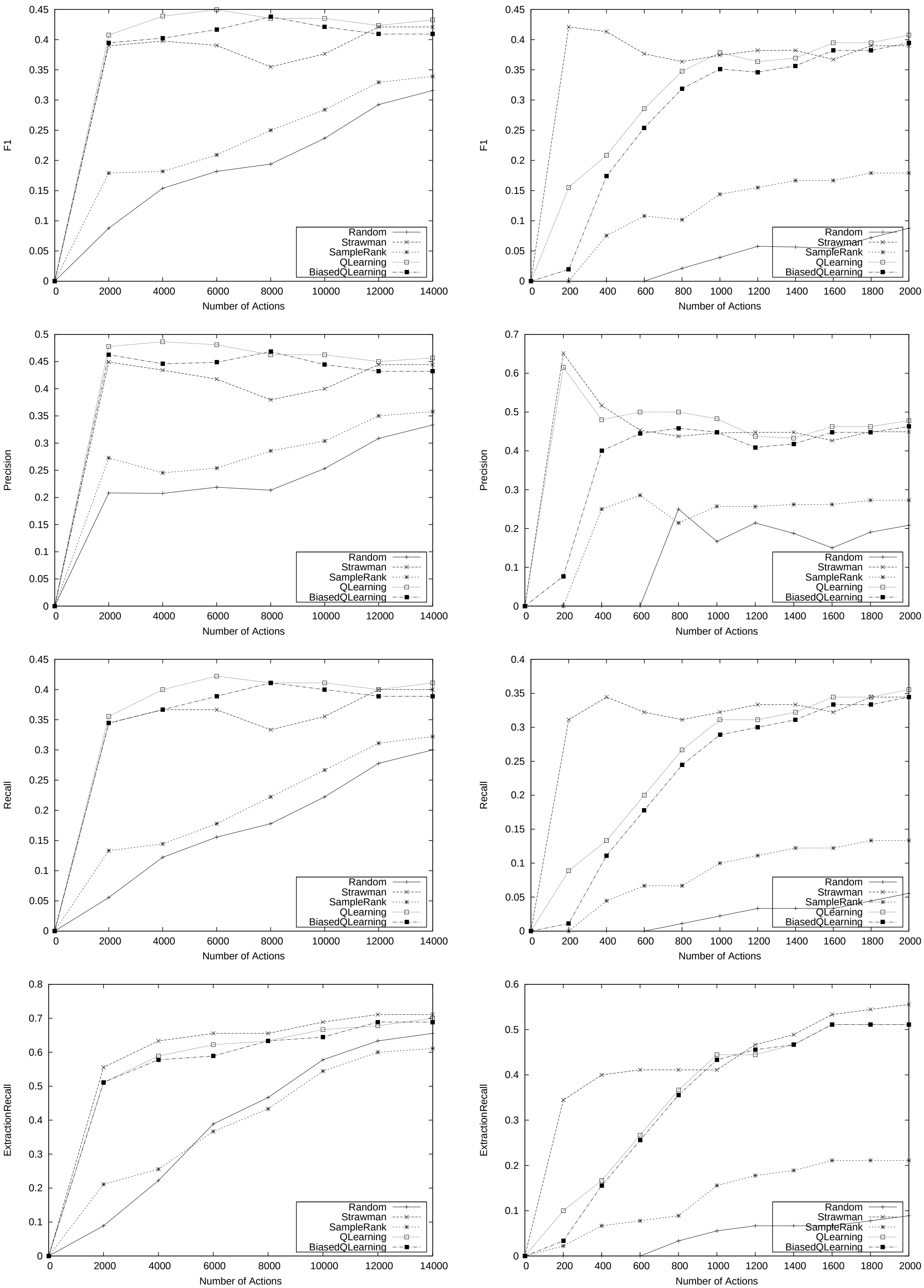


Figure 6.4. RBIE using extraction model on total entries for FindGuru task. The graphs from top to bottom are : F1, Precision, Recall and Extraction Recall

One of the key insights gained from these experiments, is that it is extremely helpful to examine all the information available at each time step in the information gathering process. This includes the results of the action taken in the previous time step, as well as all the intermediate information acquired up to that point. Such dynamically adapting, learning based approach can result in a flexible solution that can outperform strong domain-specific heuristics, like the straw-man method in our examples, and can be valuable in the absence of such domain heuristics. The success of such a learning based approach can lead to its application in many resource-conscious, real-world domains.

6.8 Chapter Summary

In this chapter, we formulated the problem of RBIE for the Web as a Markov Decision Process, and proposed the use of temporal difference q-learning to solve it. We also compare it to a fast, online, error-driven training method called SampleRank [92]. We learn a policy for effectively selecting information-gathering actions, leading to significant reduction in resource-usage. Using two challenging, real-world applications, we demonstrate that the q-learning-based approach for selecting information-gathering actions outperforms both, a random and a strong straw-man baselines. Both q-learning and SampleRank approaches effectively beat the baselines in the case of finding faculty directory URLs for computer science departments. Note that in this case, we 'bake in' the delayed reward into the objective function, making SampleRank an effective method for learning the value function. On our example task of extracting faculty email, job titles and department names, the q-learning based approach is able to achieve 88.8% of the final F1, by only using 8.6% of the total actions demonstrating its effectiveness. On this task, we find that SampleRank performs better than the random approach, but suffers due to its inability to model delayed reward.

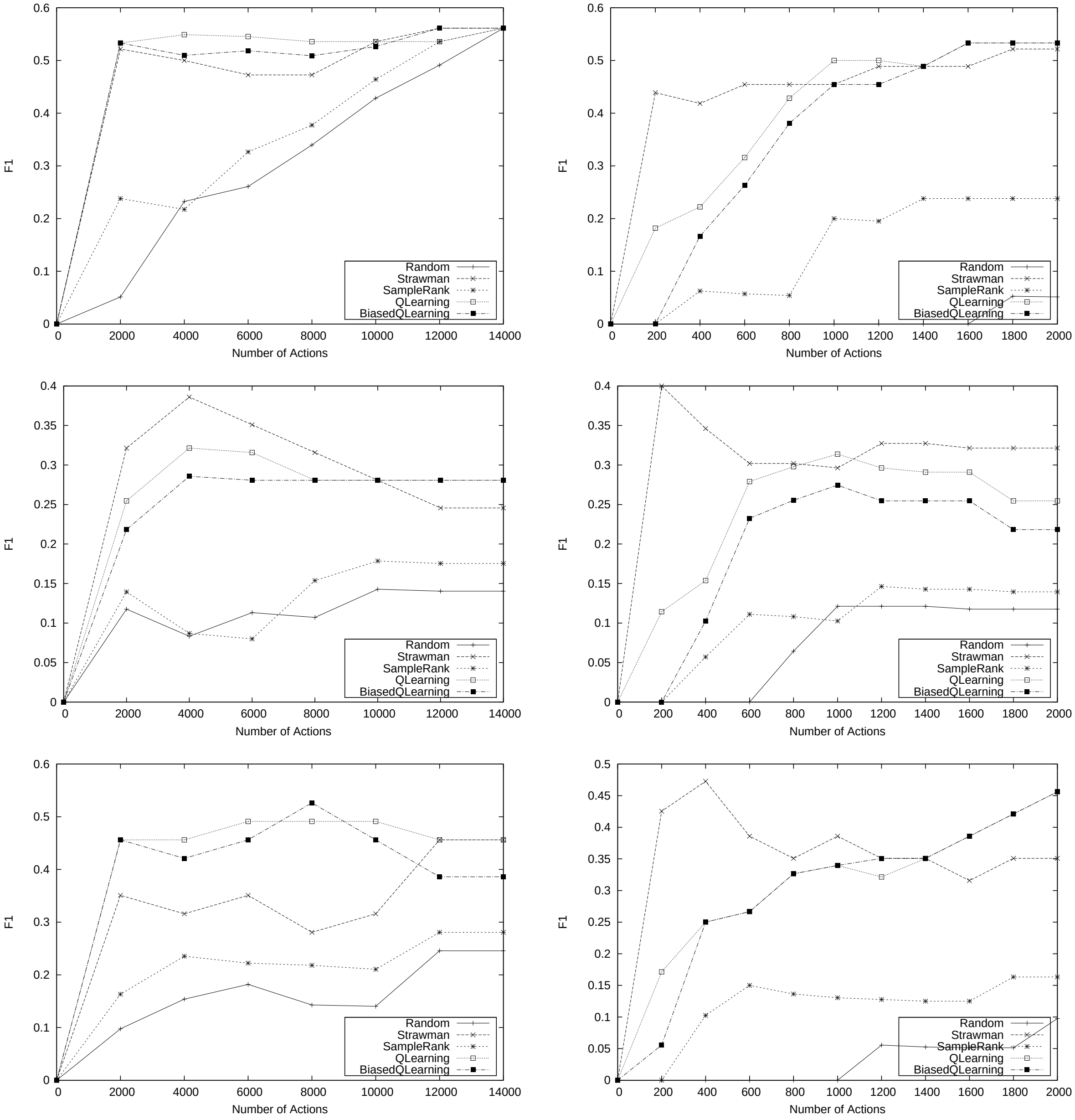


Figure 6.5. RBIE using extraction model for FindGuru task. The graphs from top to bottom are : Email, Job Title and Department Name. (figures on the right zoom to the first 2000 actions)

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this thesis, I study various aspects of Resource-bounded Information Acquisition, including selecting a subset of data for acquiring external information, exploiting interdependency in the input data for better resource utilization, and a general framework for efficiently extracting specific pieces of information from a large, external text corpus, such as the Web.

I give a specific definition of the RBIA problem, which helps develop new solutions to an important class of problems. I also answer the central question of my thesis, namely, is it possible to significantly reduce the resource requirements for acquiring external information in real-world RBIA problem domains? Using examples of special cases of the RBIA problem and extensive experiments, I demonstrate that it is possible to acquire a large fraction of the total benefit from new information, by only using a small fraction of the resources. For example, by using a reinforcement learning based framework for the task of extracting information about faculty from the Web, I show that we can obtain 88.8% of the final F1 value (that we would have been able to obtain by using all possible resource-consuming actions), by only using 8.6% of the total actions.

The Markov Decision Process based framework proposed in this thesis is general, dynamically adapting, and holistic. I now discuss various directions for improving or expanding this framework.

The first advantage of the RBIE framework is its adaptability to changing actions. We can extend the existing actions by adding new actions, modifying the existing ones, or splitting and merging them as required. For example, the experiments for our problem domain show that when extracting information for multiple fields, such as emails, job titles, and department affiliation, different acquisition strategies perform differently. Hence, one of the potential improvements to our system would be to define multiple *extract* actions, one for each field, rather than a single one that combines them all. We can also experiment with nested actions, in which the *extract* actions are nested within the corresponding *download* actions. Similarly, all the actions instantiated as a result of a *query* action are nested within it. This may also alleviate the problem of modeling delayed rewards for SampleRank, and lead to a more efficient, and fast training approach.

Our focus in this thesis is on resource-constraints at test time. However, the temporal difference approach is somewhat resource consuming to train for some domains, and it would be desirable to develop a faster training method. To this end, apart from the use of nested actions, experimenting with other variations of SampleRank, such as different sampling, parameter update, and state exploration strategies may be fruitful. One of the difficulties in training value function as described here, is that the objective function needs to be hand designed. This may not be feasible, or effective in some problem domains. Developing methods to 'learn' the reward function can be an interesting area to explore. We also presented a novel extraction technique that can scale well for large scale information gathering tasks, and supports the iterative nature of resource-bounded information acquisition. It would be interesting to study how more sophisticated extraction methods perform on similar tasks.

The basic formulation of RBIE as an MDP opens up many interesting avenues of research. Use of TD q-learning is one of the first attempts to learn general information gathering policies. More advanced techniques from the reinforcement learning literature, such as SARSA or least-square policy iteration can be explored. Currently, the proposed methods assume infinite horizon decision problem. Instead, we can apply budgeted reasoning style methods, that assume finite horizon setting.

The proposed framework is extendable in many ways. We can extend the set of information gathering actions defined here to suit the specific needs of a problem, while using the general MDP framework. Based on the requirements of the domain, we can adapt a more specific cost model, in which some actions are more expensive than others. We may also include the problem of selecting a source, for scenarios in which multiple different sources of external information are available. Furthermore, when deploying such a system for a real-world application, we need to analyze the tradeoffs between resources required for acquiring information, and the computational cost of information acquisition strategies. Another dimension to explore is the possibility for information acquisition in parallel, which may lead to interesting new paradigms.

EPILOGUE

Over the years, as I have presented this work at various venues, I have often been asked one question (in different forms): In today's world of 'Big Data', with easy availability of massive computational resources, and scalable, parallel, distributed computing platforms, is there really a need for resource-bounded information acquisition strategies? Why not just apply all our experience of working on the 'Web scale' to the problem? My answer has been this: RBIA is a fundamentally different problem, in that the information we seek from an external source is very specific. In most cases, even with the existing computational power, it would be really difficult to justify the use of sophisticated, yet computationally expensive extraction methods on 'Web scale' data, simply to complete an application-specific database. In some cases, the external information needs to be purchased at a high cost, motivating the need for accurately estimating the value of information. Also, not every individual or organization has access to such computational, or financial resources. Finally, imagination inspires us to do more with the data than what is computationally feasible. Hence, at least for the foreseeable future, there will be a need for methods that make efficient use of the available resources to achieve a task. As machine learning and data mining applications become more ubiquitous in everyday life, I hope that this thesis makes a strong case for researchers and practitioners to invest more efforts in this direction.

BIBLIOGRAPHY

[1] Agichtein, Eugene, and Gravano, Luis. Querying text databases for efficient information extraction. In *In Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE* (2003), pp. 113–124.

[2] Agrawal, Sanjay, Chakrabarti, Kaushik, Chaudhuri, Surajit, and Ganti, Venkatesh. Scalable ad-hoc entity extraction from text collections. In *PVLDB*.

[3] Arnt, Andrew, and Zilberstein, Shlomo. Learning policies for sequential time and cost sensitive classification. In *Proceedings of the KDD Workshop on Utility-Based Data Mining* (Chicago, Illinois, 2005).

[4] Attenberg, Josh, Melville, Prem, and Provost, Foster. Guided feature labeling for budget-sensitive learning under extreme class imbalance. In *ICML Workshop on Budgeted Learning*.

[5] Bansal, N., Chawla, S., and Blum, A. Correlation clustering. *In The 43rd Annual Symposium on Foundations of Computer Science (FOCS)* (2002), 238–247.

[6] Bardak, Ulas, Fink, Eugene, Martens, Chris R., and Carbonell, Jaime G. Scheduling with uncertain resources: Elicitation of additional data. In *In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* (2006).

[7] Bilgic, Mustafa, and Getoor, Lise. Voila: Efficient feature-value acquisition for classification. In *AAAI* (2007), AAAI Press, pp. 1225–1230.

[8] Bilgic, Mustafa, and Getoor, Lise. Effective label acquisition for collective classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2008), pp. 43–51. Winner of the KDD'08 Best Student Paper Award.

[9] Bilgic, Mustafa, Mihalkova, Lilyana, and Getoor, Lise. Active learning for networked data. In *ICML* (2010), pp. 79–86.

[10] Blum, Avrim, Jackson, Jeffrey, Sandholm, Tuomas, Zinkevich, Martin, Bennett, Kristin, and Cesa-bianchi, Nicol. Preference elicitation and query learning. *Journal of Machine Learning Research 5* (2004), 2004.

[11] Boutilier, Craig. A pomdp formulation of preference elicitation problems.

[12] Brin, Sergey. Extracting patterns and relations from the world wide web. In *Selected papers from the International Workshop on The World Wide Web and Databases* (London, UK, 1999), Springer-Verlag, pp. 172–183.

[13] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E.R. Hruschka, and Mitchell, T.M. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)* (2010), AAAI Press, pp. 1306–1313.

[14] Chen, Fei, Doan, Anhai, Yang, Jun, and Ramakrishnan, Raghu. Efficient information extraction over evolving text data. In *in ICDE* (2008).

[15] Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. *ML 15*, 2 (1994), 201–221.

[16] Craig Boutilier, Kevin Regan, and Viappiani, Paolo. Online feature elicitation in interactive optimization. In *26th International Conference on Machine Learning*.

[17] Craig Boutilier, Kevin Regan, and Viappiani, Paolo. Simultaneous elicitation of preference features and utility. In *AAAI*.

[18] Culotta, Aron. *Learning and inference in weighted logic with application to natural language processing*. PhD thesis, University of Massachusetts, 2008.

[19] Dalvi, Bhavana, Cohen, William, and Callan, Jamie. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *Web Search and Data Mining*.

[20] Daskalakis, Constantinos, Karp, Richard M., Mossel, Elchanan, Riesenfeld, Samantha, and Verbin, Elad. Sorting and selection in posets. *CoRR abs/0707.1532* (2007).

[21] DeRose, Pedro, Shen, Warren, Chen, Fei, Lee, Yoonkyong, Burdick, Douglas, Doan, AnHai, and Ramakrishnan, Raghu. Dblife: A community information management platform for the database research community (demo). In *CIDR* (2007), www.crdrdb.org, pp. 169–172.

[22] Dong, Xinyi, Halevy, Alon Y., Nemes, Ema, Sigurdsson, Stefan B., and Domingos, Pedro. Semex: Toward on-the-fly personal information integration. In *Workshop on Information Integration on the Web (IIWEB)* (2004).

[23] Donmez, Pinar. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *In Proceedings of CIKM08* (2008).

[24] Donmez, Pinar, Carbonell, Jaime G., and Schneider, Jeff. Efficiently learning the accuracy of labeling sources for selective sampling. In *In Proc. of the 15th ACM SIGKDD international* (2009).

[25] Druck, Gregory, Mann, Gideon, and McCallum, Andrew. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2008), pp. 595–602.

[26] Eliassi-Rad, Tina. *Building Intelligent Agents that Learn to Retrieve and Extract Information.* PhD thesis, University of Wisconsin, Madison, 2001.

[27] Esmeir, Saher, and Markovitch, Shaul. Anytime algorithms for learning resource-bounded classifiers. In *ICML Workshop on Budgeted Learning* (2010).

[28] Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D., and Yates, A. Web-scale information extraction in knowitall. In *Proceedings of the International WWW Conference, New York* (May 2004), ACM.

[29] Etzioni, O., Hanks, S., Jiang, T., Karp, R. M., Madani, O., and Waarts, O. Efficient information gathering on the internet (extended abstract), 1996.

[30] Etzioni, Oren, Fader, Anthony, Christensen, Janara, and Soderl, Stephen. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence.*

[31] Gatterbauer, Wolfgang. Estimating required recall for successful knowledge acquisition from the web. In *Proceedings of the 15th international conference on World Wide Web* (New York, NY, USA, 2006), WWW '06, ACM, pp. 969–970.

[32] Gatterbauer, Wolfgang. Rules of thumb for information acquisition from large and redundant data. Tech. Rep. arXiv:1012.3502, Dec 2010.

[33] Golovin, Daniel, Krause, Andreas, and Ray, Debajyoti. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, Eds. 2010, pp. 766–774.

[34] Grass, J., and Zilberstein, S. A value-driven system for autonomous information gathering. *Journal of Intelligent Information Systems 14* (2000), 5–27.

[35] Grishman, Ralph, and Sundheim, Beth. Message understanding conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics* (Morristown, NJ, USA, 1996), Association for Computational Linguistics, pp. 466–471.

[36] Han, Hui, Zha, Hongyuan, and Giles, Lee. Name disambiguation in author citations using a k-way spectral clustering method. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL 2005)* (2005).

[37] Heckerman, D., Horvitz, E., and Middleton, B. An approximate nonmyopic computation for value of information. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 3 (1993), 292–298.

[38] Huang, Jian, and Yu, Cong. Prioritization of domain-specific web information extraction. In *In AAAI* (2010).

[39] Jenny Rose Finkel, Trond Grenager, and Manning, Christopher. Incorporating non-local information into information extraction systems by gibbs sampling.

[40] Josh Attenberg, Prem Melville, Foster Provost, and Saar-Tsechansky., Maytal. Selective data acquisition. In *Cost-Sensitive Machine Learning*, B. Krishnapuram, S. Yu, and R.B. Rao, Eds. Chapman and Hall, 2011.

[41] Kanani, Pallika, and McCallum, Andrew. Resource-bounded information gathering for correlation clustering. In *Computational Learning Theory 07, Open Problems Track, COLT 2007* (2007), pp. 625–627.

[42] Kapoor, Aloak, and Greiner, Russell. Learning and classifying under hard budgets. In *ECML* (2005), pp. 170–181.

[43] Kapoor, Ashish, and Horvitz, Eric. Breaking boundaries: Active information acquisition across learning and diagnosis.

[44] Knoblock, C. A. Planning executing, sensing and replanning for information gathering. In *IJCAI* (1995).

[45] Krause, Andreas, and Guestrin, Carlos. Near-optimal nonmyopic value of information in graphical models. In *Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI* (2005), p. 05.

[46] Kuwadekar, Ankit, and Neville, Jennifer. Combining semi-supervised learning and relational resampling for active learning in network domains.

[47] Lafferty, John, McCallum, Andrew, and Pereira, Fernando. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML* (2001), Morgan Kaufmann, pp. 282–289.

[48] Lesser, Victor R., Horling, Bryan, Klassner, Frank, Raja, Anita, Wagner, Thomas, and Zhang, Shelley XQ. Big: An agent for resource-bounded information gathering and decision making. *Artif. Intell 118* (2000), 197.

[49] Lewis, David D., and Catlett, Jason. Heterogeneous uncertainty sampling for supervised learning. In *ML94* (San Francisco, CA, jul 1994), MKP, pp. 148–156.

[50] Li, Liuyang, Pczos, Barnabs, Szepesvri, Csaba, and Greiner, Russ. Budgeted distribution learning of belief net parameters, 2010.

[51] Lin, J., Fernandes, A., Katz, B., Marton, G., and Tellex, S. Extracting answers from the web using knowledge annotation and knowledge mining techniques, 2002.

[52] Linh, Ta Nha. Harvesting useful information from researchers home pages. Tech. rep., 2009.

[53] Lizotte, Dan, Madani, Omid, and Greiner, Russell. Budgeted learning of naive-Bayes classifiers. In *UAI03* (Acapulco, Mexico, 2003).

[54] Macskassy, Sofus A. Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In *Proceedings of the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009).

[55] Malewicz, Grzegorz. Parallel scheduling of complex dags under uncertainty. In *SPAA* (2005), pp. 66–75.

[56] Massachusetts, Harald Steck, Steck, Harald, and Jaakkola, Tommi S. Unsupervised active learning in large domains, 2002.

[57] McCallum, A., and Li, W. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL* (2003), pp. 188–191.

[58] McCallum, Andrew, and Wellner, Ben. Object consolidation by graph partitioning with a conditionally-trained distance metric. *KDD Workshop on Data Cleaning, Record Linkage and Object Consolidation* (2003).

[59] McCallum, Andrew Kachites. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[60] McCallum, Andrew Kachites. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[61] Melville, Prem, Rosset, Saharon, and Lawrence, Richard D. Customer targeting models using actively-selected web content. In *KDD* (2008), Ying Li, Bing Liu, and Sunita Sarawagi, Eds., ACM, pp. 946–953.

[62] Melville, Prem, Saar-Tsechansky, Maytal, Provost, Foster, and Mooney, Raymond. Active feature-value acquisition for classifier induction. In *ICDM04* (2004), pp. 483–486.

[63] Melville, Prem, Saar-Tsechansky, Maytal, Provost, Foster, and Mooney, Raymond. An expected utility approach to active feature-value acquisition. In *Proceedings of the International Conference on Data Mining* (Houston, TX, November 2005), pp. 745–748.

[64] Melville, Prem Noel. *Creating diverse ensemble classifiers to reduce supervision.* PhD thesis, Austin, TX, USA, 2005. AAI3217133.

[65] Murphy, Kevin P. Active learning of causal bayes net structure. Tech. rep., 2001.

[66] Nagy, Istvn, Farkas, Richrd, and Jelasity, Mrk. Researcher affiliation extraction from homepages.

[67] Nagy, Istvn T., and Farkas, Richrd. Person attribute extraction from the textual parts of web pages. In *In Third Web People Search Evaluation Forum (WePS-3), CLEF 2010* (2010).

[68] Nakashole, Ndapandula, Theobald, Martin, and Weikum, Gerhard. Find your advisor: robust knowledge gathering from the web. In *Procceedings of the 13th International Workshop on the Web and Databases* (New York, NY, USA, 2010), WebDB '10, ACM, pp. 6:1–6:6.

[69] Nath, Aniruddh, and Domingos, Pedro. A language for relational decision theory.

[70] Nodine, Marian H., Fowler, Jerry, Ksiezyk, Tomasz, Perry, Brad, Taylor, Malcolm, and Unruh, Amy. Active information gathering in infosleuth. *International Journal of Cooperative Information Systems 9*, 1-2 (2000), 3–28.

[71] Padmanabhan, Balaji, Zheng, Zhiqiang, and Kimbrough, Steven O. Personalization from incomplete data: what you don't know can hurt. In *KDD01* (2001), pp. 154–163.

[72] Rattigan, Matthew J., Maier, Marc, and Jensen, David. Exploiting network structure for active inference in collective classification. *Data Mining Workshops, International Conference on 0* (2007), 429–434.

[73] Rennie, Jason, and McCallum, Andrew. Efficient web spidering with reinforcement learning. In *Proceedings of the International Conference on Machine Learning* (1999).

[74] Roy, Nicholas, and McCallum, Andrew. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning* (2001), Morgan Kaufmann, pp. 441–448.

[75] Roy, Nicholas, and McCallum, Andrew. Toward optimal active learning through sampling estimation of error reduction. In *ML01* (2001), Morgan Kaufmann, San Francisco, CA, pp. 441–448.

[76] Russell, Stuart, and Norvig, Peter. *Artificial Intelligence: A Modern Approach (2nd Edition).* Prentice Hall, 2003.

[77] Saar-Tsechansky Maytal, Melville Prem, Provost Foster. Active information acquisition for model induction. In *Management Science* (2008).

[78] Settles, Burr. Active learning literature survey. Tech. rep., 2010.

[79] Shchekotykhin, Kostyantyn, Jannach, Dietmar, and Friedrich, Gerhard. xcrawl: a high-recall crawling method for web mining. *Knowl. Inf. Syst. 25* (November 2010), 303–326.

[80] Sheng, Victor, Provost, Foster, and Ipeirotis, Panagiotis G. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2008), ACM, pp. 614–622.

[81] Sheng, Victor S., and Ling, Charles X. Feature value acquisition in testing: a sequential batch test algorithm. In *ICML '06: Proceedings of the 23rd international conference on Machine learning* (New York, NY, USA, 2006), ACM, pp. 809–816.

[82] Suchanek, Fabian, Sozio, Mauro, and Weikum, Gerhard. SOFIE: A self-organizing framework for information extraction. In *Proceedings of the 18th World Wide Web Conference (WWW 2009)* (Madrid, Spain, 2009), Association for Computing Machinery (ACM), ACM, p. ?

[83] Sutton, Richard S., and Barto, Andrew G. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[84] Tang, Jie, Zhang, Jing, Yao, Limin, Li, Juanzi, Zhang, Li, and Su, Zhong. Arnetminer: Extraction and mining of academic social networks. In *Knowledge Discovery and Data Mining.*

[85] Thompson, C. A., Califf, M. E., and Mooney, R. J. Active learning for natural language parsing and information extraction. In *ICML* (1999), p. 406.

[86] Tong, Simon, and Koller, Daphne. Active learning for parameter estimation in bayesian networks. In *In NIPS* (2001), pp. 647–653.

[87] Turney, Peter. Types of cost in inductive concept learning. In *In Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning* (2000), pp. 15–21.

[88] Viappiani, Paolo, and Boutilier, Craig. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23 (NIPS)* (2010).

[89] Vijayanarasimhan, S., and Kapoor, A. Visual recognition and detection under bounded computational resources. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (june 2010), pp. 1006 –1013.

[90] Watkins, Christopher J. C. H., and Dayan, Peter. Q-learning. *Machine Learning 8*, 3-4 (1992), 279–292.

[91] Whitelaw, Casey, Kehlenbeck, Alex, Petrovic, Nemanja, and Ungar, Lyle. Web-scale named entity recognition. In *Proceedings of the 17th ACM conference on Information and knowledge management* (New York, NY, USA, 2008), CIKM '08, ACM, pp. 123–132.

[92] Wick, Michael, Rohanimanesh, Khashayar, Bellare, Kedar, Culotta, Aron, and McCallum, Andrew. Samplerank: Training factor graphs with atomic gradients. In *ICML* (2011).

[93] Wu, Fei, Hoffmann, Raphael, and Weld, Daniel S. Information extraction from wikipedia: moving down the long tail. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2008), ACM, pp. 731–739.

[94] Xindong Wu, Kui Yu, Wang, Hao, and Ding, Wei. Online streaming feature selection. In *ICML*.

[95] Yang, Liu, Carbonell, Jaime, Yang, Liu, and Carbonell, Jaime. Cost-reliability tradeoff, 2009.

[96] Yao, Limin, Tang, Jie, and Li, Juanzi. A unified approach to researcher profiling. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence* (Washington, DC, USA, 2007), WI '07, IEEE Computer Society, pp. 359–366.

[97] Zhao, Shubin, and Betz, Jonathan. Corroborate and learn facts from the web. In *KDD* (2007), pp. 995–1003.

[98] Zheng, Z., and Padmanabhan, B. On active learning for data acquisition. In *Proceedings of IEEE International Conference on Data Mining* (2002).

[99] Zilberstein, Shlomo. Resource-bounded reasoning in intelligent systems. *ACM Comput. Surv 28* (1996).

[100] Zilberstein, Shlomo, and Lesser, Victor. Intelligent information gathering using decision models. *Technical Report 96-35, Computer Science Department University of Massachusetts at Amherst* (1996).